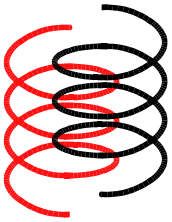


Koehn
Consulting

Beratung in betriebswirtschaftlichen Anwendungen

Synactive

GuiXT / InputAssistant / Viewer / Controls

Dokumentation

30.Mai 2012, G. Koehn

© by Synactive GmbH

INHALTSVERZEICHNIS

1. GRUNDSÄTZLICHES ZU GUIXT 5

1.1. WAS LEISTET GUIXT ? 5
1.2. TECHNOLOGIE 5
1.3. EIN KLEINES BEISPIEL 5
1.4. WIE ERFASSE UND ÄNDERE ICH EIN GUIXT SKRIPT? 5
1.5. WO SIND DIE GUIXT SCRIPTE ABGELEGT ? 5
1.6. UNTER WELCHEM NAMEN SIND DIE GUIXT SCRIPTE ABGELEGT ? 6
1.7. FORMAT DER SCRIPTANWEISUNG 7
1.8. WIE SPRICHT MAN EINZELNE BILDELEMENTE AN ? 7
1.9. POSITIONIERUNG 8
1.10. RELATIVE POSITIONIERUNG 8
1.11. VARIABLEN IN SCRIPTEN 8
1.12. TIPS UND TRICKS 9

2. HINWEISE ZUR INSTALLATION 10

2.1. GENERELL 10
2.2. INSTALLATION FÜR GROSSE BENUTZERZAHLEN 13
2.3. GUIXT PROFILE 14
2.4. ABLAGE DER PROFILE PARAMETER 25
2.5. BENUTZERKATALOG 28
2.6. GUIXT SCRIPT ABLAGE IM MIME REPOSITORY 30
2.7. RFC (REMOTE FUNCTION CALL) IN GUIXT 32
2.8. MEHRSPRACHENFÄHIGKEIT 35
2.9. USERSTATISTICS 38

3. GUIXT BEDIENUNG 44

3.1. DAS GUIXT FENSTER UND EDITOR 44
3.2. FEHLERSUCHE 47
3.3. SYSTEM TRACE 49
3.4. DEBUGGER 52
3.5. GUIXT SCRIPT EDITOR 55

4. BEFEHLE IM DETAIL (GUIXT) 62

4.1. BOX 62
4.2. BOXSIZE 62
4.3. BUTTONSIZE 63
4.4. COLUMNHEADER 63
4.5. COLUMNORDER 64
4.6. COLUMNSIZE 64
4.7. COLUMNWIDTH 65
4.8. COMMENT 65
4.9. COMPRESS 66
4.10. DEFAULT 66
4.11. DEL 67
4.12. DISABLE 69
4.13. FIELDSIZE 70
4.14. FILE 70
4.15. GLOBALTEXTREPLACE 70
4.16. ICON 71
4.17. IF / ELSE / ENDIF 72
4.18. IGNOREWARNING 76
4.19. IMAGE 76
4.20. INCLUDE 82
4.21. LISTIMAGE 82
4.22. MARK 85
4.23. MESSAGE 87
4.24. NOAREARESIZE 89
4.25. NODROPDOWNLIST 89
4.26. NOINPUT 89

4.27. NoLeadingZeros ... 90
4.28. NoScrollbar ... 90
4.29. Numerical ... 90
4.30. Offset ... 91
4.31. PlaySound ... 91
4.32. ProcessingOption ... 92
4.33. Pos ... 95
4.34. Pushbutton ... 96
4.35. Radiobutton ... 99
4.36. Required ... 100
4.37. Stop ... 100
4.38. TableWidth ... 101
4.39. Text ... 101
4.40. TextReplace ... 102
4.41. Tip ... 103
4.42. Title ... 103
4.43. TitlePrefix ... 104
4.44. TitleSuffix ... 104
4.45. Uppercase ... 105
4.46. VersionNumber ... 105
4.47. View ... 106
4.48. Windowposition ... 111
4.49. Windowsize ... 111

5. SYSTEMVARIABLEN ... 112

6. INPUT ASSISTANT ... 117

6.1. Was leistet der InputAssistant ? ... 117
6.2. Eingabehistorie ... 117
6.3. Lokale Wertehilfe ... 119
6.4. InputSet's direkt im R/3 Bild ... 124
6.5. InputScripts ... 128
6.6. Einbildtransaktion ... 130
6.7. Lokale Wertehile und Matchcode für eigene Eingabefelder ... 133
6.8. InputRecording ... 139

7. DIE BEFEHLE IM DETAIL (INPUTASSISTANT) ... 140

7.1. Appendfile ... 140
7.2. ApplyGuiScript ... 140
7.3. Call ... 141
7.4. Checkbox ... 146
7.5. Checkdate ... 146
7.6. ClearCallCache ... 147
7.7. Closefile ... 147
7.8. Column ... 148
7.9. CompareText ... 149
7.10. ConnectHtml ... 150
7.11. Converttext ... 154
7.12. CopyFile ... 155
7.13. CopyText ... 155
7.14. Decimalseparator ... 157
7.15. DropDownList ... 158
7.16. Enter ... 160
7.17. GetFieldAttribute ... 162
7.18. GetTableAttribute ... 163
7.19. Goto/Label ... 163
7.20. HistoryUpdate ... 164
7.21. Inputfield ... 164
7.22. Leave ... 170
7.23. Link ... 170
7.24. Localvaluehelp ... 172
7.25. On ... 172
7.26. Openfile ... 173

7.27. PARAMETER 174
7.28. RADIOBUTTON 175
7.29. READFILE 176
7.30. REMOVEFILE 177
7.31. RENAMEFILE 177
7.32. RETRY ON ERROR 177
7.33. REPLACETEXT 178
7.34. RETURN 178
7.35. RETURNVALUES 179
7.36. SEARCHHELP 180
7.37. SELECTDIRECTORY 181
7.38. SELECTFILE 181
7.39. SCREEN 182
7.40. SET 183
7.41. SETACTIVETAB 187
7.42. SETCURSOR 187
7.43. SETLEFTTAB 189
7.44. SORT 189
7.45. START 190
7.46. STATUSMESSAGE 190
7.47. TAB 191
7.48. TABLE 192
7.49. TEXTBOX 195
7.50. USING 196

8. GUIXT CONTROLS 197

8.1. FUNKTIONSÜBERSICHT 197
8.2. INSTALLATION / KONFIGURATION 197

9. DIE BEFEHLE IM DETAIL (GUIXT CONTROLS) 198

9.1. CONTROL 198
9.2. CALLVBS 199
9.3. CALLVBSASYNC 202
9.4. OBJECT GUIXT IN VBS SCRIPT 202

10. VIEWER 207

10.1. FUNKTIONSÜBERSICHT 207
10.2. INSTALLATION / KONFIGURATION 207
10.3. VIEWHELP 207
10.4. HELPVIEW VERZEICHNISSE 209

11. SPECIAL TOPICS 211

11.1. DATEN AUS LISTEN LESEN 211
11.2. TABELLEN IN GUIXT UND INPUTASSISTANT 215
11.3. TABELLEN IM AUFRUF VON RFC-FUNKTIONSBAUSTEINEN (CALL) 219
11.5. LISTEN IN HTML-FORMAT ANZEIGEN (CALL) 228
11.6. LANGTEXTE 235
11.7. VARIABLEN, BERECHNUNGEN, VERGLEICHSOPERATIONEN 239
11.8. EXTERNE DATENQUELLEN INTEGRIEREN 246
11.9. DIE BEHANDLUNG VON DATEIEN IN INPUTSCRIPTS 249
11.10. LÖSCHEN VON MENUEINTRÄGEN 253
11.11. SAPGUI SCRIPTING UND GUIXT 258
11.12. AUFNEHMEN VON EIGENEN SPALTEN IN TABLECONTROL 262

1. Grundsätzliches zu GUIXT

1.1. Was leistet GuiXT ?

Mit GuiXT können Sie R/3 Bilder vereinfachen und an die Erfordernisse einzelner Benutzergruppen anpassen, ohne R/3 Programme oder Bilder modifizieren zu müssen.

Neben der Vereinfachung der Bilder bietet GuiXT eine Reihe von Möglichkeiten, die den Benutzer angenehmer und schneller mit R/3 arbeiten lassen: Sie können zum Beispiel Eingabefelder durch Auswahlknöpfe ersetzen, Abbildungen und Hilfetext einblenden, Feldtexte unternehmensspezifisch gestalten, zusätzliche Drucktasten definieren,...

1.2. Technologie

GuiXT arbeitet mit einer einfachen Skriptsprache. Pro R/3 Bild können Sie in einem Skript das gewünschte Layout in Form einzelner Anweisungen definieren.

1.3. Ein kleines Beispiel

```
del    [Umrechnungsdat]
pos    [Erste Belegposition] (8,0)
```

Das Eingabefeld mit der Bezeichnung `Umrechnungsdat` wird gelöscht, die Feldgruppe `Erste Belegposition` wird auf Zeile `8`, Spalte `0` verschoben.

1.4. Wie erfasse und ändere ich ein GuiXT Skript?

Gehen Sie im R/3 System auf das Bild, das Sie verändern möchten. Klicken Sie dann im GuiXT Menü auf **Edit->Skript**. Es wird dann der Texteditor (Notepad) für das entsprechende Skript gestartet.

1.5. Wo sind die GuiXT Scripte abgelegt ?

Die GuiXT Skripte sind als normale ASCII-Text-Dateien abgelegt, und zwar entweder unmittelbar auf Ihrem PC oder auf einem Fileserver im lokalen Netz. Sie legen das Verzeichnis, in dem die Skripte liegen, im GuiXT-Profile guixt.ini durch den Parameter zwar entweder unmittelbar auf Ihrem PC oder auf einem Fileserver im lokalen Netz. Sie legen das Verzeichnis, in dem die Skripte liegen, im GuiXT-Profile guixt.ini durch den Parameter
Directory1 directoryname fest, z.B. **Directory1** C:\GuiXT\Script1
Insgesamt können Sie 4 Verzeichnisse **Directory1,...,Directory4** angeben. Sie werden jeweils in der Reihenfolge **Directory4,...,Directory1** nach einem passenden Skript durchsucht, das erste gefundene Skript wird ausgeführt.
Beispielsweise könnte **Directory1** auf ein Verzeichnis mit unternehmensweit definierten Skripten zeigen, die auf einem Fileserver liegen, und **Directory2** könnte spezielle Skripte für eine kleinere Benutzergruppe enthalten.
Über das GuiXT Profile können die Skripte unterschiedlich pro Benutzer oder Benutzergruppe zugeordnet werden.
Falls Sie sich in mehr als einem R/3-System oder in verschiedenen Mandanten anmelden und hierfür unterschiedliche Skriptverzeichnisse verwenden wollen, können Sie das durch Systemvariablen in den Namen der Skriptverzeichnisse erreichen:
&client wird ersetzt durch den aktuellen R/3 Mandanten
&database wird ersetzt durch die aktuelle Datenbanknummer (System Id)
&user wird ersetzt durch den aktuellen Benutzer
Beispiele:

Directory1	\\Server\GuiXT\scripts
Directory2	F:\GuiXT\scripts\&database
Directory3	C:\GuiXT\myscripts\&database

Wir arbeiten z.B. in dem R/3 System TST. Als erstes wird GuiXT nach dem entsprechenden Skript in dem als Directory3 angegebenen Verzeichnis C:\GuiXT\myscripts\TST suchen. Wird das Skript dort nicht gefunden, sucht GuiXT im Verzeichnis F:\GuiXT\scripts\TST und schließlich im Verzeichnis \\Server\GuiXT\scripts.
Im R/3 Logon-Fenster wird die Variable &client durch den Wert 000 ersetzt, da der vom Benutzer gewählte Mandant noch nicht bekannt ist.

1.6. Unter welchem Namen sind die GuiXT Scripte abgelegt ?

Jedes GuiXT Skript bezieht sich auf genau ein R/3 Bild (Dynpro). Im R/3 Repository sind die Bilder unter dem ABAP Programmnamen und einer 4-stelligen Nummer abgelegt, z.B. Dynpro 0100 zum Programm SAPMF05A. Da die GuiXT Skripte sprachabhängig sind (Feldnamen, Texte, Bilder), kommt noch der Sprachenschlüssel hinzu, z.B. E für Englisch, D für Deutsch, F für Französisch, I für Italienisch, J für Japanisch,... Der Name eines Skriptfiles setzt sich damit insgesamt wie folgt zusammen:

pppppppp.Lnnnn.txt

pppppppp:	ABAP Programmname
L:	Sprachenschlüssel D, E, F, I, J, ... (1-stelliger Sprachenschlüssel aus dem R/3 System)
nnnn:	4-stellige Dynpronummer

Beispiel:

SAPMF05A.D0100.txt
Hiervon gibt es zwei Ausnahmen:

- Das Skript zum R/3 logon-Bild wird unter dem Namen **Dlogon.txt, Elogon.txt,**... abgelegt: Der Sprachenschlüssel D, E,... ist dabei die Sprache, in der das logon-Bild angezeigt wird, da die vom Benutzer gewählte Sprache noch nicht bekannt ist
- Die Bereichsmenüs sind unter dem Namen des jeweiligen GUI-Programms **MENUxxxx** abgelegt (statt **SAPMSYST**), um die Bereichsmenüs jeweils unterschiedlich gestalten zu können. Zum Beispiel: **MENUS000.D0040.txt** oder **MENUPR00.D0040.txt**

1.7. Format der Scriptanweisung

- Jede Skript-Anweisung steht in einer separaten Zeile
- Leere Zeilen werden überlesen
- Kommentare in einer Zeile können durch // eingeleitet werden
- Das Skipt-Schlüsselwort steht immer am Anfang, z.B. `pos`
- Bildelemente sind in eckige Klammern eingeschlossen, zum Beispiel `[Umrechnungsdat]`

Bildkoordinaten sind in runde Klammern eingeschlossen, zum Beispiel `(8,0)`

1.8. Wie spricht man einzelne Bildelemente an ?

Bildelemente sind

- Felder
- Feldgruppen
- Drucktasten (Pushbutton)
- Auswahlknöpfe (Radiobutton)
- Ankreuzfelder (Check box)
- Table Controls
- Tabstrip Controls
- R/3 Controls
- Menüeinträge

Sie werden über ihren Text, in eckige Klammern eingeschlossen, angesprochen, zum Beispiel `[Umrechnungsdat]` oder `[Kontonummer]`.

Zu beachten: Den Text müssen Sie so eingeben, wie er in der Anzeige erscheint, bis auf folgende mögliche Abweichungen:

- Groß-/Kleinschreibung ist beliebig, zum Beispiel `[KontoNummer]` oder `[kontonummer]` statt `[Kontonummer]`
- Zwischenräume sind beliebig, zum Beispiel `[Konto Nummer ]` statt `[KontoNummer]`
- Zur Abkürzung langer Bezeichnungen können Sie einen Punkt am Ende verwenden, z.B. `[Erste Belegpos.]` oder `[Erste B.]` statt `[Erste Belegposition]`
- Bei Drucktasten ohne Text (nur Ikone) können Sie den Quickinfo-Text der Drucktaste verwenden

Wahlweise kann zusätzlich der Typ des Bildelements in Form eines Kennbuchstabens angegeben werden, etwa wenn eine Feldgruppe und ein Feld mit dem gleichen Namen existieren. Folgende Typen sind möglich:

- `F[...]` Feld
- `G[...]` Feldgruppe
- `P[...]` Drucktaste (Pushbutton) innerhalb des Bilds oder in der Drucktastenleiste
- `R[...]` Auswahlknopf (Radiobutton)
- `C[...]` Ankreuzfeld (Check box)
- `S[...]` Tabstrip Control
- `T[...]` Table Control
- `X[...]` R/3 Control (z.B. Image Control)
- `M[...]` Menüeintrag

Beispiel:

```
del F[Umrechnungsdat]
pos G[Erste Belegposition] (8,0)
```

Falls in Sonderfällen ein Bildelement ohne Text angesprochen werden muß, kann das durch folgende Schreibweise geschehen:

- `#[x,y]`
  wobei `x` die Zeile und `y` die Spalte (jeweils bei 0 beginnend) des Bildelements ist.
- für Table Controls ohne Bezeichnung: `[table]` oder `T[table]`

Falls es mehrere Felder gleichen Namens gibt, z.B. zweimal `[KontoNummer]`, dann wird das erste Feld (von links oben nach rechts unten gezählt) mit dem normalen Namen `[KontoNummer]` angesprochen, die folgenden mit `[KontoNummer.2]`, `[KontoNummer.3]`, ...

Falls es mehrere Table Controls ohne Namen gibt, wird der erste mit `[table]` oder `T[table]` der folgende: `[table.2]` oder `T[table.2]`, ...

1.9. Positionierung

Zur Angabe der Position `(zeile,spalte)` auf dem Bild gibt es drei Möglichkeiten:

- `(x,y)` Hierbei ist `x` die Zeile, `y` die Spalte. Zum Beispiel bedeutet `(4,60)`: Zeile `4`, Spalte `60`. Die Zählung beginnt bei `(0,0)` falls nicht durch `offset` ein anderer Nullpunkt festgelegt wurde. `x` und `y` sind ganzzahlige positive Werte. Nur bei zwei Anweisungen (`image` und `view`) können `x` und `y` auch Dezimalzahlen (mit Dezimalpunkt) sein, zum Beispiel `(1.5,2.8)`

- `[name]` Angabe eines Bildelements, dessen Position hier eingesetzt wird

- `[name]+(x,y)` Angabe eines Bildelementsplus einem Offset in Zeilen und Spalten, das zur Position des Bildelements hinzuaddiert wird. Für x und y sind hierbei auch negative Werte erlaubt, zum Beispiel bedeutet `[KontoNummer]+(1,-10)` eine Zeile unter und 10 Spalten links von dem Feld `KontoNummer`

1.10. Relative Positionierung

Um z.B. eine Gruppe von Radiobuttons als Ganzes hin- und herschieben zu können, kann man durch ein "Offset"-Kommando ein Offset in der Form `(zeile,spalte)` angeben, das automatisch zu allen folgenden direkt angegebenen Koordinaten addiert wird.

Gleichbedeutend sind z.B.

```
Offset       (10,21)
Radiobutton (0,0) "..."
Radiobutton (1,0) "..."
Radiobutton (2,0) "..."
```

und

```
Radiobutton (10,21) "..."
Radiobutton (11,21) "..."
Radiobutton (12,21) "..."
```

1.11. Variablen in Scripten

In allen Strings `"..."` einer Skript-Kommandozeile können ein oder mehrere Variablen eingefügt werden, die sich auf Felder des aktuellen Bildes oder eines vorangegangenen Bilds beziehen können. Sie werden automatisch ersetzt durch den jeweiligen Feldinhalt.
Die Bezeichnung der Variablen setzt sich zusammen aus dem &-Zeichen gefolgt von der Bezeichnung des entsprechenden Felds in eckigen Klammern, z.B. `&[Kontonummer]`.
Beispiel:

```
Title "Anzeige Konto &[Kontonummer]"
```

Falls der aktuelle Inhalt des Felds `[KontoNummer]` 113100 war, ist das Skript-Kommando äquivalent mit
`Title "Anzeige Konto 113100"`

Ab und zu ist es nötig, auch die Eingabewerte aus vorher prozessierten Bildern zur Verfügung zu haben. Beispielsweise könnte eine Materialnummer im Einstiegsbild der Transaktion eingegeben worden sein und Sie benötigen sie im zweiten Bild zum Zusammensetzen des Namens einer Produktabbildung, die Sie anzeigen möchten. Sie können in diesem Fall durch die Angabe von zwei Doppelpunkten :: vor dem Namen der Variablen (innerhalb der eckigen Klammer) ausdrücken, daß es sich um ein Eingabefeld auf einem vorangegangenen Bild handelt.
Beispiel:

```
Image (4,80) "P:\products\img&[::Materialnummer].jpg"
```

Wurde beispielsweise die Materialnummer 00500187 auf dem Einstiegsbild eingegeben, so wird nun das Bild `P:\products\img00500187.jpg` angezeigt.

Als systemseitig vordefinierte Variable steht daneben das Tagesdatum `&[today_...]` in verschiedenen Formaten zur Verfügung, wobei man die Buchstaben d=Tag, m=Monat und y=Jahr frei zusammensetzen kann.

Beispiele für das Datum 24. Dezember 1999:

`&[today_dmy]`	24121999
`&[today_mdy]`	12241999
`&[today_y]`	1999
`&[today_m]/&[today_y]`	12/1999
`&[today_user]`	24.12.1999 (userspezifische Aufbereitung, je nach Grundeinstellung User)

Ebenfalls unterstützt ist die Angabe einer Anzahl Tage, die zum Tagesdatum dazuaddiert oder davon subtrahiert wird. Beispiele für das Datum 24. Dezember 1999:

```
&[today+1_dmy] 25121999
&[today-1_dmy] 23121999
&[today+8_d.m.y] 01.01.2000

&
```

1.12. Tips und Tricks

- GuiXT muß bereits vor dem Start des R/3 Sapgui aktiviert sein. Sie können das bequem dadurch erreichen, daß Sie statt saplogon.exe ein **saplogon.bat** mit Ihrem Saplogon Desktop Icon verbinden, in dem zunächst guixt.exe aufgerufen wird und anschließend saplogon.exe gestartet wird:

  Start C:\guixt\guixt.exe
  Start C:\sapgui\saplogon.exe

- Durch den Parameter **StartHidden Yes** in guixt.ini kann optional für einzelne Benutzer GuiXT völlig unsichtbar ablaufen. Eine Alternative ist **StartMinimized Yes**
- Falls eine Skript-Anweisung anscheinend keine Wirkung hat, liegt das vielleicht an einer falsch geschriebenen Bezeichnung für ein Bildelement, zum Beispiel haben Sie `[Umrechnungsdatum]` statt `[Umrechnungsdat]` geschrieben.
  Wenn Sie in GuiXT das Skript-Fenster anzeigen , werden entsprechende Syntaxfehlermeldungen in die Anzeige des aktuellen Skripts eingeblendet. Klicken Sie hierzu auf **View->Skript.**

Statt des manuellen Erfassens des Skripts kann auch ein R/3 add-on Tool, der **GuiXT Designer**, eingesetzt werden, um das Layout in einem WYSIWYG-Editor zu definieren. Näheres hierzu finden Sie auf den Webseiten von Synactive unter http://www.synactive.com.

2. Hinweise zur Installation

2.1. Generell

Die Installation der GuiXT Komponenten ist relativ einfach. Je nach Benutzerzahl können Sie sich zwischen lokaler Installation an jedem Arbeitsplatz oder zentraler Installation entscheiden. Die Hardware-Anforderungen an den PC unterscheiden sich nicht von den normalen Sapgui-Anforderungen.

Eine Installation an einem einzelnen Arbeitsplatz zum Test und zur Entwicklung von Skripten nimmt nur wenige Minuten in Anspruch. Sie finden im folgenden konkrete Hinweise zur Vorgehensweise.

Die folgenden Programme müssen für GuiXT vorhanden sein:

- guixt.exe
- guixt.dll
- pinnacle.dll (nur falls GuiXT Viewer für html und rtf-Dateien eingesetzt wird)

Die Gesamtgröße der drei Programme liegt unter 3 Megabyte.

Ab **Sapgui 4.6B** müssen die Programme in dem sapgui-Verzeichnis liegen (z.B. C:\Programme\SAPPc\Sapgui), also in dem gleichen Verzeichnis, in dem z.B. auch front.exe liegt. Falls Sie Sapgui von einem zentralen Netzwerkserver starten, werden die GuiXT Programme dort ebenfalls zentral abgelegt.

Zur Installation in früheren Sapgui-Ständen können Sie die GuiXT Programme in einem beliebigen Verzeichnis lokal oder zentral ablegen. Bei der Nutzung an vielen PCs empfiehlt sich eine zentrale Ablage, um einen Upgrade der Version einfach durchführen zu können.

> Für eine Installation **bei einer großen Zahl von Benutzern,** die keine Skripte verändern sollen, ist eine reine Server-Installation bequemer. GuiXT kann zum produktiven Einsatz vollständig (Programme, Skripte und Profile) auf einem Netzwerk-Server installiert werden. Auf den einzelnen Benutzer-PCs sind dann weder Programme noch Registry-Einträge noch GuiXT-Skripte nötig. Detaillierte Hinweise finden Sie auf der nächsten Seite.

Schrittweise Installationsanleitung

1 Download der Programmdateien	Laden Sie **GuiXT Setup** auf Ihren PC. Informationen über die aktuelle Version erhalten Sie über das Version Info auf der Download Seite. Der Download GuiXT Setup umfasst die GuiXT Programmdateien einschließlich der Komponenten InputAssistant und Viewer.
2 Setup ausführen	Führen Sie die Datei guixtsetup.exe aus. Das Setup schlägt Ihnen als Installationsverzeichnis C:\GuiXT vor.
3 Programmdateien nach SapGui kopieren	Falls Sie Sapgui **4.6B** oder höher einsetzen, so kopieren Sie die GuiXT Programme (guixt.exe und guixt.dll, für Viewer auch pinnacle.dll) bitte in Ihr **sapgui Directory**. Im allgemeinen ist dies **C:\Programme\sappc\sapgui**. Bitte stellen Sie im Fall von Sapgui 4.6B auch sicher, dass **Sapgui Patch 31** eingespielt ist (front.exe build 1484). **Update** Falls Sie keine Neuinstallation vornehmen, sondern eine Update-Version von

GuiXT/InputAssistant/Viewer installieren, ist der Vorgang hiermit abgeschlossen und das Update steht ohne weiteres zur Verfügung.

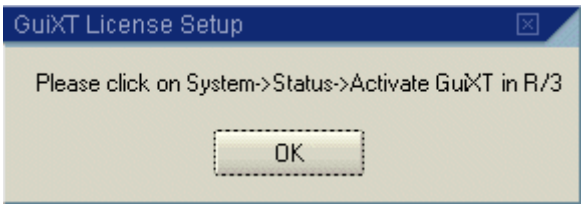
4 Lizenzschlüssel anfordern

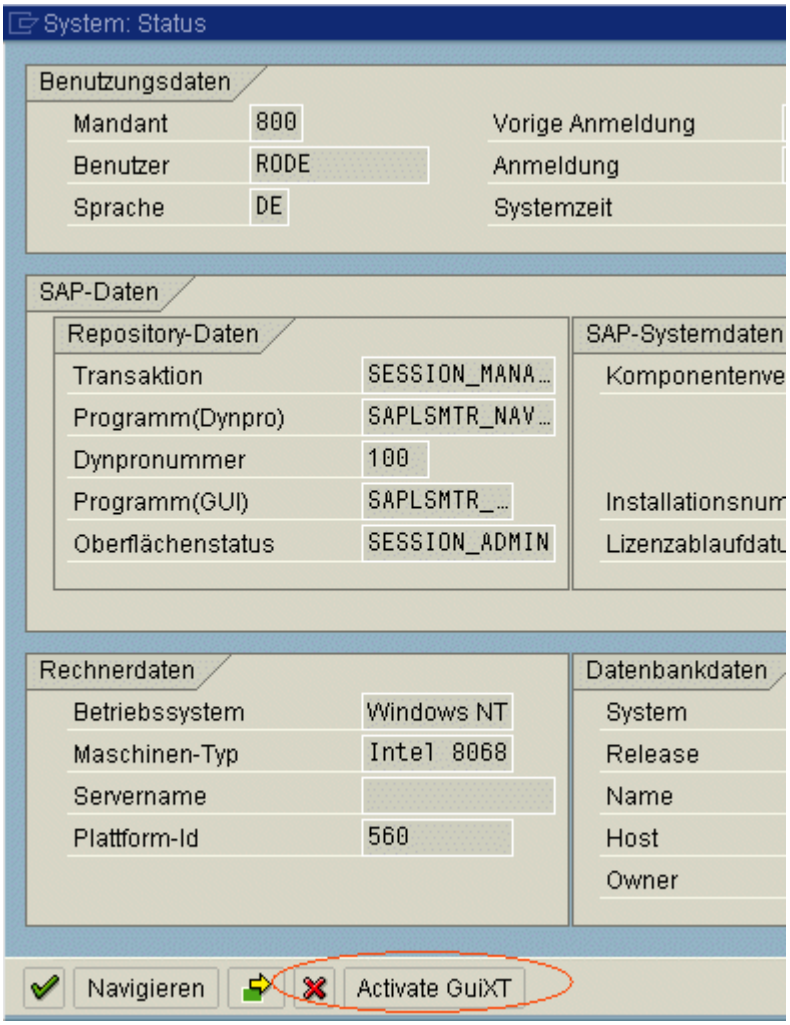
Notieren Sie Ihre SAP Installationsnummer; sie wird angezeigt, wenn Sie in R/3 auf System->Status klicken. Beispiel: 0220452822.
Füllen Sie nun das Formular zur Anforderung des Testkeys aus. Synactive schickt Ihnen dann über email, meist innerhalb der nächsten 30 Minuten, die Lizenzschlüssel zurück; Testkeys sind 30 Tage lang gültig.
Beispiel:
GuiXT-Lizenzschlüssel (mit InputAssistant, Viewer): xrgbhzbq

5 GuiXT und R/3 aufrufen

Wenn Sie Sapgui **4.6B** oder höher einsetzen, wird GuiXT automatisch beim Starten des R/3 Systems ebenfalls gestartet, wenn Sie GuiXT in den Sapgui Optionen aktiviert haben:

Möglicherweise müssen Sie beim ersten Aufruf im SAP Menü "System Status" den Button "GuiXT aktivieren" am unteren Fensterrand betätigen.





Falls Sie Sapgui vor **4.6B** einsetzen, starten Sie GuiXT vor dem Aufruf von R/3.

6 GuiXT Profile einrichten, GuiXT Verzeichnisse einrichten

Zeigen Sie über Edit->Profile in GuiXT das GuiXT Profile an und tragen Sie den GuiXT-Testschlüssel dort ein (Feld Testkey). Erstellen bzw. wählen Sie ein Verzeichnis, in dem Sie Ihre Skripte ablegen wollen (z.B. C:\GuiXT\Skripte) und tragen Sie dieses Verzeichnis unter "Directory1" im GuiXT Profile ein. Tragen Sie ggf. auch die Verzeichnisse für History und Domains ein.

7 Designer installieren

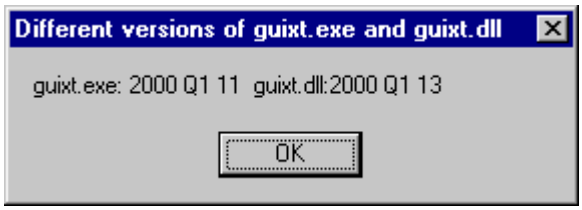
Laden Sie **Designer Setup** auf Ihren PC und führen Sie das Setup-Programm aus. Fordern Sie den Lizenzschlüssel an. Starten Sie aus dem GuiXT Fenster heraus über Tools->Designer den GuiXT Designer und tragen Sie dort den Designer Lizenzschlüssel über die Menüpunkte Tools->License manager ein.

8 Installation der Dokumentation/Hilfefunktion und der Tutorials

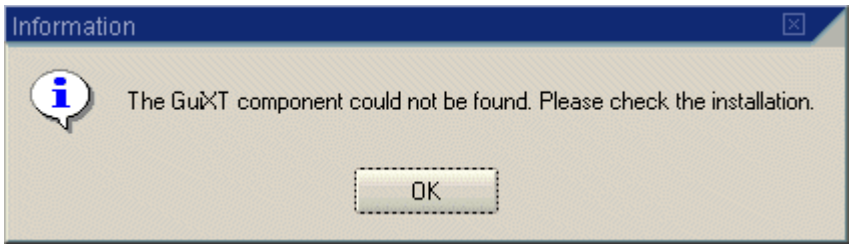
Laden Sie die html Version der Dokumentation und der Tutorials in der von Ihnen präferierten Sprache von der Download Seite. Entpacken Sie die Dateien. Geben Sie als Verzeichnis das von Ihnen gewählte GuiXT Verzeichnis an, standardmäßig ist dies C:\GuiXT. Die entpackten Dateien werden in Unterverzeichnissen abgelegt und stehen sogleich über die Hilfefunktion von GuiXT zur Verfügung.

Mögliche Probleme nach der Installation

Bitte achten SIe bei einem eventuellen Kopieren der Programme von einem Verzeichnis in ein anderes darauf, daß Sie keine unterschiedlichen Versionen mischen. Andernfalls erfolgt beim Anmelden in R/3 eine Fehlermeldung:



Falls die GuiXT-Programme für Sapgui 4.6b nicht im Sapgui-Verzeichnis vorhanden sind, der Benutzer aber in den sapgui-Options GuiXT aktiviert, so erfolgt eine Fehlermeldung:



Beachten Sie bitte auch die Infos auf den Q&A Seiten.

2.2. Installation für grosse Benutzerzahlen

Sie haben die Skripte nun lokal entwickelt und sie bei einigen ausgesuchten Benutzern testen lassen. Jetzt soll eine grössere Benutzerzahl mit der neuen Oberfläche arbeiten. Was ist zu beachten?

(1)
Sind die GuiXT Programme (guixt.exe, guixt.dll, eventuell für den Viewer auch pinnacle.dll) in der richtigen Version auf den Benutzerarbeitsplätzen installiert?

Neue GuiXT-Versionen kommen mit den sapgui-Patches mit. Es ist auch immer möglich, eine neuere GuiXT Version (immer guixt.exe und guixt.dll gleichzeitig) mit einem älteren sapgui-Stand einzusetzen. Sie können zu einer sapgui-Installation auch eigene Dateien (z.B. pinnacle.dll oder guixt.ini) hinzufügen und sie damit verteilen, siehe SAP Hinweis 512040. Vielleicht sind aber auch andere Tools zur Softwareverteilung bei Ihnen im Einsatz. Stellen Sie nach Möglichkeit sicher, dass alle Benutzer mit der gleichen Version arbeiten, und dass diese Version auch Grundlage der Tests Ihrer Skripte war.

Das GuiXT Setup Programm kann auch automatisch, ohne Benutzerdialoge, ablaufen. Einzelheiten finden Sie in "Unbeaufsichtigte Installation".

(2)
Wo werden die Skripte abgelegt, und wie werden spätere Aktualisierungen vorgenommen?

Im allgemeinen ist eine zentrale Ablage der Skripte sinnvoll. Hierfür hat sich das SAP Web Repository bewährt. Sie können jedoch auch einen zentralen Fileserver oder http-Server verwenden, wenn dieser mit hoher Sicherheit auch verfügbar ist. Zur Entlastung des Netzwerks ist die Einschaltung des GuiXT Cache sinnvoll. Falls Sie das Web Repository als Ablage verwenden, können Sie durch Einsatz der "VersionNumber" eine weitere Optimierung der Zugriffe erreichen.

(3)
Einstellung des GuiXT Profiles auf den Benutzerarbeitsplätzen

Falls in Ihrem Unternehmen ein Tool eingesetzt wird, mit dem Sie zentral die Registry-Einträge der Benutzer-PCs pflegen können, so ist die Ablage des GuiXT Profiles in der Registry eine akzeptable Möglichkeit. In der Regel wird jedoch vorgezogen, mit einer lokalen Datei guixt.ini zu arbeiten, die über "SwitchTo" auf eine zentrale ini-Datei verweist. Die lokale ini-Datei enthält also die Einträge

```
IgnoreRegistry  Yes
SwitchTo        \\server\.....\guixt.ini  //zentrale ini-Datei
```

Sie können dann spätere Änderungen der Profile-Parameter einfach in der zentralen ini-Datei vornehmen.

Bei der Pflege der ini-Datei bitte an folgende Punkte denken: "StartHidden Yes" um das GuiXT Fenster unsichtbar zu lassen, "IgnoreRegistry Yes" um lokale Profile-Einträge zu ignorieren, eventuell Einschalten des GuiXT Cache.

(4)
Information der Benutzer, Vorstellung der geänderten Abläufe

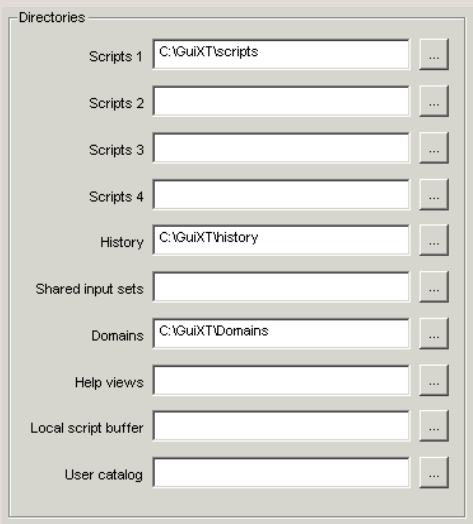
Wenn auch Ihre neue Oberfläche sicherlich einfacher zu bedienen ist als der SAP-Standard, so bedeutet es für die Benutzer doch eine Umstellung. Es ist sicher nicht falsch, einen kleinen Leitfaden und eine kurze Schulung dazu anzubieten.

(5)
Fortlaufende Verbesserungen einplanen

In der Regel haben die Benutzer noch eine Menge weiterer Ideen, was alles schneller und einfacher gehen könnte, und einige davon werden Sie in der Zukunft wahrscheinlich noch umsetzen. Es ist empfehlenswert, diesen Prozess von Anfang an einzuplanen und die entsprechenden Abläufe zu definieren (Sammeln der Ideen, Entscheidung über Aufnahme in Projekt, Skript-Entwicklung, Versionsverwaltung, Test, Rollout) .

2.3. GuiXT Profile

Allgemeines	Durch Klicken auf **Edit->Profile** im GuiXT Fenster können Sie die Konfigurationsparameter anzeigen oder ändern. GuiXT unterstützt die Ablage der Profile-Parameter sowohl in der Windows Registry als auch in einer guixt.ini Datei. Näheres dazu finden Sie in Ablage der Profile Parameter `Für die Verwendung in der guixt.ini-Datei finden Sie hier jeweils den Parameternamen`



Directories
(Verzeichnisse)

Hier können Sie verschiedene Verzeichnisse angeben. Diese können sich lokal auf Ihrem eigenen PC oder in einem lokalen Netzwerk befinden. Sowohl Laufwerkbuchstaben als auch UNC-Notation (\\servername\directory\...) sind möglich.

Sie können Systemvariablen benutzen, um die Verzeichnisse abhängig von SAP-System (&database), Mandanten (&client) oder Benutzer (&winuser) zu gestalten (z.B.: Directory1 \\Server\GuiXT\scripts\&database). Ebenfalls möglich sind Windows Umgebungsvariablen in der Form &%[ENVNAME], wobei ENVNAME der Name der Umgebungsvariablen ist, also z.B. C:\&%[GUIXTDIR]\Scripts.

Scripts1,... **Scripts4** sind die Verzeichnisse in denen nach GuiXT-Skripten gesucht wird. Die Verzeichnisse werden immer in der Reihenfolge Scripts4, dann Scripts3, dann Scripts2, dann Scripts1 durchsucht. Sie könnten zum Beispiel, firmenweite Skripte in Scripts1 und eigene Skripte in Scripts2 ablegen. Speziell während der Skript-Entwicklungsphase kann es interessant sein, ein persönliches Verzeichnis für Skripte vorübergehend zu erstellen. Sie können die angegebenen Verzeichnisse mit sofortiger Wirkung aus- und einschalten.

GuiXT Skripte, Bilder (gif, jpg) und InputScripte können Sie ablegen in:

- lokalen Dateien, z.B. unter C:\GuiXT\Scripts
- zentralen Fileservern, z.B. \\p00145\GuiXT\Scripts
- http-Servern, z.B. http://www.synactive.com/demo.d.45
- ftp-Servern, z.B. ftp://synactive.com/demo.d.45
- SAP Web Repository (SAP Datenbank), z.B. sapwr:z.guixt.demo
  Näheres hierzu finden Sie in: GuiXT Skripte im SAP Web Repository

```
Directory1
Directory2
Directory3
Directory4
```

Historywird vom InputAssistant benutzt. Hier werden die Werte der Eingabehistorie abgelegt und es können private Input Sets dort definiert werden.
Bei der Eingabehistorie würde ein zentrales Verzeichnis dazu führen, daß die Eingabewerte aller Benutzer gemischt werden, was sicher nicht sinnvoll ist. Hier ist also auf jeden Fall, auch bei zentralen Programmen, Skripten und Profile, die Angabe eines lokalen Verzeichnisses nötig, z.B. C:\GuiXT\History. Der Benutzer sollte aus Datenschutzgründen darüber informiert sein, daß und wo die Historie seiner Eingabewerte abgelegt ist.

```
History
```

Shared Input Sets wird vom InputAssistant benutzt. Hier können Input Sets abgelegt werden, die mehreren Benutzern zur Verfügung stehen sollen.

```
InputSets
```

Domains beinhalten die Domäne-Dateien für die lokale Wertehilfe (InputAssistant).

```
Domains
```

Help views Verzeichnis mit eigenen html-Hilfedateien (Viewer). Pro Transaktion kann in dem angegebenen Verzeichnis eine Datei abgelegt werden, z.B. d.transaction.VA01.html für Transaktion VA01, Sprache D. Zusätzlich muss das Verzeichnis eine Datei help.gif enthalten (Format 18x18 Pixel, z.B.) , die von GuiXT als Ikone rechts in der toolbar angezeigt wird, falls zu der aktuellen Transaktion eine entsprechende Hilfedatei existiert. Durch eine eigene Image-Anweisung mit ViewHelp=... kann die angegebene Datei im GuiXT Skript übersteuert werden.
Im SAP Logon-Bild wird die Datei d.transaction.logon.html gesucht, wobei der Sprachenschlüssel gemäß der Sprache des Logon-Bildes eingesetzt wird.

```
HelpViewDirectory
```

Local script buffer. Dieses Verzeichnis enthält Skripte und Bilder, die von einem zentralen Fileserver gelesen wurden und hier zum schnelleren Zugriff lokal abgelegt sind. Siehe auch Profile-Parameter **File Cache.**
Bei Verwendung von Transaktionsvarianten mit GuiXT Skripten werden die aus der zentralen SAP Datenbank gelesenen Skripte und Bilder ebenfalls hier abgelegt. Sie werden nach einer Änderung in SAP automatisch aktualisiert.

```
Directory0
```

User catalog. Hier können Sie eine Textdatei angeben, in der benutzerbezogen unterschiedliche Skriptverzeichnisse definiert werden. Nähere Informationen finden Sie in der Beschreibung Benutzerkatalog.

```
UserCatalog
```

File cache

Bei Aktivieren der Option File Cache werden alle von einem zentralen Server im Netzwerk gelesenen Dateien (Skripte, Bilder, Wertehilfe,...) lokal abgespeichert, um einen schnelleren Zugriff zu gewährleisten.

- Alle Dateien aus einem Verzeichnis mit UNC-Notation (mit \\ beginnend, z.B. \\p00145\GuiXT\Scripts) werden lokal gepuffert.
- Bei Verwendung eines Laufwerkbuchstabens, z.B. S:, erfolgt dagegen keine Pufferung, auch wenn der Buchstabe einem externen Verzeichnis zugeordnet ist.
- Jeweils bei dem ersten Zugriff auf eine gepufferte Datei pro SAP Session wird das Änderungsdatum der Datei auf dem Server geprüft und die Datei falls nötig neu vom Server geladen.
- Danach erfolgen innerhalb einer SAP Session alle Zugriffe auf die Datei aus dem lokalen Cache.

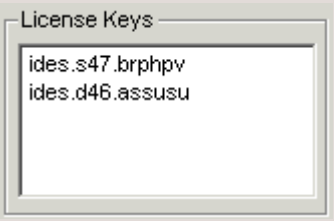
Wenn Sie Skripte von einem Webserver (ftp/http) einlesen, ist es für eine optimale Performance empfehlenswert, File Cache zu aktivieren.

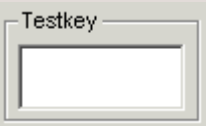
`Replication Yes/No`

Encryption

Alle Skripte, InputScripte und der "User Catalog" werden im Cache in verschlüsselten Format zwischengespeichert.

`CacheEncryption Yes/No`



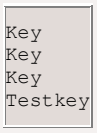


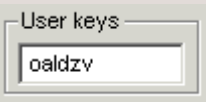
License keys and Testkey
(Lizenz- und Testschlüssel)

Sie benötigen einen Lizenzschlüssel um die Synactive Komponente einzusetzen. Allein die Komponente GuiXT, ohne irgendeine weitere Komponente, kann ohne Lizenzschlüssel eingesetzt werden.

Bitte beachten Sie, daß Lizenzschlüssel abhängig von Ihrer Komponentenwahl, Ihrer SAP Installationsnummer sowie der Namen Ihrer Systeme sind.

Testkeys können auf Anfrage durch Synactive, für eine Testdauer von 30 Tagen, erstellt werden. Die Testschlüssel sind für alle Synactive Komponenten (ausser dem Designer) gültig. Bitte verwenden Sie die Internetseite "Testkey" um einen Testschlüssel zu beantragen. Dieser wird per email zugeschickt.

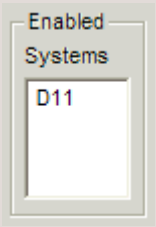




User keys
(Benutzerschlüssel)

Ein Benutzerschlüssel wird generiert wenn Sie auf "Activate GuiXT" im SAP Status-Bildschirm klicken. Dies brauchen Sie nur einmal, nach Installation eines neuen Lizenzschlüssels, durchzuführen.

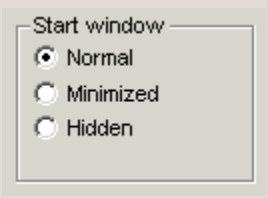
Diese Schlüssel benötigen keine manuelle Änderung.



Enabled Systems

Sie können die Namen von SAP Systemen angeben für die die GuiXT-Konfiguration aktiviert werden soll. Den Namen des Systems finden Sie unter "System" "Status" "Datenbankdaten" "Name", z.B. D11. Sie können mehrere Systeme angeben.

```
EnableSystem XXX
EnableSystem YYY
```



Start window

(Start-Fenster)

Beim Start von GuiXT, kann das GuiXT-Fenster sichtbar ("visible"), minimiert und sichtbar ("minimized") oder versteckt ("hidden") sein. Der Modus "hidden" (versteckt) ist besonders sinnvoll für Anwender, die keine Änderung an den Skripten oder an den Einstellungen durchzuführen haben.

Bitte beachten Sie, daß bei aktivierten "Hidden"-Option, der Konfigurationsbildschirm auch nicht mehr angezeigt wird. Wenn Sie GuiXT erneut starten mit Angabe des Parameters "visible", haben Sie erneut Zugriff auf das GuiXT-Profil und können die Konfigurationseinstellungen sehen und evtl. ändern:

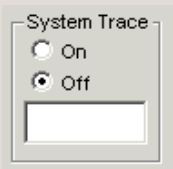
guixt.exe visible

```
StartMinimized Yes/No
StartHidden Yes/No
```

Icon Id

Diese Option bezieht sich auf den Befehl **Tip** in GuiXT Skripten. Sie können eine beliebige standard Icon Id aus dem SAP System angeben. Dieses Icon zeigt dem Benutzer, daß ein Tip (Quickinfo) zur Verfügung steht.

```
Icon_Tip_Field
Icon_Tip_Box
```

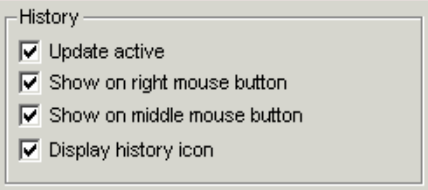


System Trace

Diese Option wird ausschließlich für Debugging-Zwecke verwendet. Wenn diese Option aktiviert ist, wird das System einen Trace in einem separaten Fenster anzeigen.

Bei aktivierter Trace Option (System Trace On) wird der Trace in der Datei gespeichert, die Sie hier benennen (ggf. mit Pfadangabe).

```
Trace Yes/No
TraceFile
```



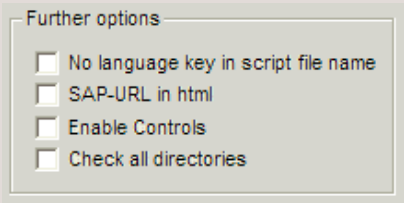
History

Diese Optionen beziehen sich auf den InputAssistant. Ist die Option **Update active** aktiviert, wird der InputAssistant die History-Dateien bei jeder Eingabe updaten.

Die beiden nächsten Optionen **Show...** steuern, ob die Eingabehistorie bei Klick auf die rechte oder mittlere Maustaste angezeigt wird. Die Applikationstaste kann immer zur Anzeige der Historie verwendet werden.

Die Option Display history icon zeigt das HistoryIcon auf Bildschirmen für die Werte in der Historie abgelegt wurden. Durch Drücken auf das HistoryIcon werden die Felder mit den zuletzt eingegebenen Werten gefüllt. Der Defaultwert ist "No".

```
HistoryUpdate Yes/No
HistoryOnRightButton Yes/No
HistoryOnMiddleButton Yes/No
HistoryShowIcon Yes/No
```



Further Options

No language key in script file name

Wenn diese Option markiert ist, sucht GuiXT nach Skripten ohne Sprachenschlüssel, z.B. anstelle der Skriptdatei SAPMF05A.D0100.txt liest es die Datei SAPMF05A.0100.txt.
Zusätzlich finden Sie im GuiXT Screen Elements Fenster die technischen Feldnamen in der ersten Spalte. Siehe Special Topic Multi language support

```
NoLanguageKey Yes/No
```

SAP-URL in html

Bei Ankreuzen dieser Option kann bei einer beliebigen html Anzeige im Internet Explorer mit der SAP URL gearbeitet werden.

```
LinkSapUrl Yes/No
```

Enable Controls

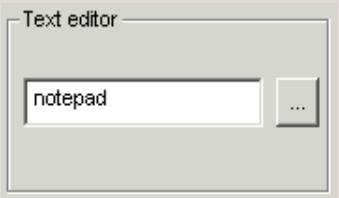
Beim Prozessieren eines InputScripts werden Controls automatisch mit aufgebaut (andernfalls nur bei "Enable Controls" im InputScript). Im allgemeinen ist es sinnvoll, diese Option einzuschalten.

```
EnableControls Yes/No
```

Check all directories

GuiXT sucht jedes Skript in allen angegebenen Verzeichnissen.

```
CheckAllDirectories Yes/No
```



Text editor

GuiXT verwendet standardmäßig das Programm Notepad als Skripteditor. In diesem Feld können Sie wahlweise einen alternativen Texteditor benennen wie zum Beispiel den kostenlosen GuiXT Editor, der speziell zur Erstellung von GuiXT Skripten von Synactive GmbH entwickelt wurde.

```
Texteditor
```

User Options

(Benutzer-Optionen)

In GuiXT Skripten können Sie diese Optionen abfragen mit dem Befehl **`if Q[Option=xxxx]`**. Sie könnten zum Beispiel eine Option **`display_help`** anbieten. Aktiviert der Benutzer diese Option im GuiXT-Profil, könnten Sie einen Hilfetext zu einer bestimmten Transaktion anzeigen, entweder eine Bild-Datei mit dem Befehl **`Image`** oder eine rtf- oder html-Datei mit dem Befehl **`View`**. Jeder Benutzer kann also entscheiden ob er den Hilfetext anzeigen lassen möchte.

```
Option
Option
Option
```

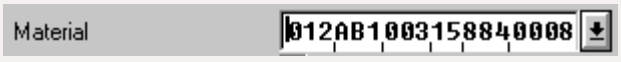
Special material number layout

(Spezielle Darstellung der Materialnummer)

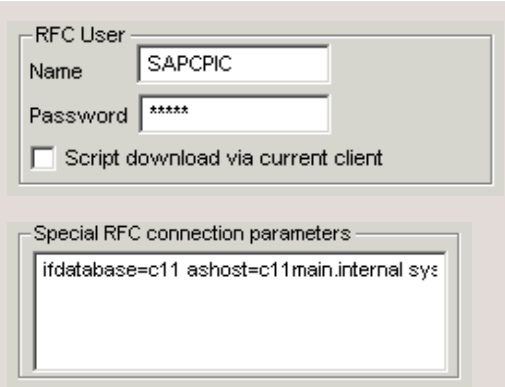
Diese spezielle Option gehört nicht zum Standard des GuiXT. Sollten Sie Interesse an dieser Option haben, nehmen Sie bitte mit Synactive GmbH Kontakt auf.

Ziel dieser Option, ist die Materialnummer in einer strukturierten Form anzuzeigen. Sie können ein beliebiges Bitmap angeben, das automatisch in dem unteren Bereich des Materialnummerneingabefeld angezeigt wird.

Beispiel:




```
Materialmask1
Materialmask2
Materialmask3
Materialmask4
```



RFC User

Unter diesem Benutzernamen werden RFC-Aufrufe durchgeführt. Das Passwort ist verschlüsselt abgelegt. Nähere Informationen finden Sie in: RFC (Remote Function Call) in GuiXT. Sie können den Rfc-Aufruf über den aktuellen Mandanten durch Ankreuzen von "Script download via current client" steuern.

```
RfcUser
RfcPassword

RfcPasswordEnc (Passwort verschlüsselt)

Hinweis: Um das RFC Passwort in verschlüssselter Form zu erhalten,
rufen Sie guixt auf mit
guixt RfcPassword <password>
wobei <password> das alte Passwort ist. Das verschlüsselte Passwort
wird dann angezeigt und gleichzeitig in das Windows Clipboard
übertragen, sodass Sie es mit Strg+V in die guixt.ini Datei einfügen
können.

RfcConnect Explizite Parametrisierung der RFC Verbindung.

Man kann entweder den Applikationsserver direkt benennen durch
ASHOST=... und SYSNR=... , oder mit load balancing (Gruppenanmeldung)
arbeiten durch MSHOST=... R3NAME=... GROUP...
Beispiel:
RfcConnect ASHOST=tst0120.internal SYSNR=00

Es ist ebenfalls möglich, mehrere Systeme zu unterscheiden:

RfcConnect ifDatabase=C11 ASHOST=c11main.internal SYSNR=00
RfcConnect ifDatabase=PR1 MSHOST=pr1msg.prod R3NAME=PR1 GROUP=PUBLIC

RfcCurrentClient Yes Rfc-Aufruf über den aktuellen Mandanten
```

Beispiel für eine typische guixt.ini Datei

```
IgnoreRegistry Yes
StartHidden Yes

// Komponenten und Lizenzschlüssel
Component GuiXT
Component InputAssistant
Key jnfglm

// Scripts und Input Sets von Server
Directory1 \\P00112\GuiXT\Scripts
InputSets \\P00112\GuiXT\Inputsets

// History
HistoryUpdate Yes
History C:\GuiXT\History

// Lokale Wertehilfe Domains
\\P00112\GuiXT\Domains

// Cache
Replication Yes
Directory0 C:\GuiXT\Temp
```

Weitere Parameter in guixt.ini

Die folgenden Parameter können nur in guixt.ini gesetzt werden:

`IgnoreRegistry Yes`

Die GuiXT-Profile-Einträge in der Windows Registry werden bei **IgnoreRegistry Yes** ignoriert. Nur die Parameter in guixt.ini sind wirksam. Empfehlenswert falls die Parametrisierung nur über guixt.ini erfolgen soll (in diesem Fall liegt die Datei guixt.ini im SAP GUI-Verzeichnis), .

`Optional No`

Sobald GuiXT in den SAP GUI-Optionen einmal eingeschaltet ist, kann es nicht mehr durch den Benutzer ausgeschaltet werden.

Bitte beachten Sie, daß diese Option im Enjoy-Modus nicht einsetzbar ist.
In diesem Fall muß der Eintrag
HKEY_CURRENT_USER/Software/SAP/SAPGUI Front/SAP Frontend Server/Administration
in der Registry geändert werden: Fügen Sie eine neue String Variable "DisappearItems" hinzu, der Sie den Wert "Activate GuiXT" geben. Falls diese Variable bereits mit anderen Einträgen vorhanden ist, dann fügen Sie den Eintrag "Activate GuiXT" am Ende hinzu, mit einem ';' abgetrennt, z.B. "Generate graphic;Activate GuiXT".
Wenn GuiXT immer aktiv sein soll, dann setzen Sie den Wert "GuiXT" in
HKEY_CURRENT_USER/Software/SAP/SAPGUI Front/SAP Frontend Server/Customize
auf "1".

`FileAccessTimeout`

Maximale Zeit für den Zugriff auf "http://...", "ftp://..." oder "\\server\..." Dateien. Beispiel:

FileAccessTimeout 60

Nach 1 Minute (60 Sekunden) versucht GuiXT stattdessen, den File aus seinem lokalen Cache zu laden. Im GuiXT Protokoll erfolgt ein Timeout-Eintrag.

Falls FileAccessTimeout nicht explizit angegeben ist, arbeitet GuiXT mit einem Defaultwert von 300 = 5 Minuten.

`RfcAccessTimeout`

Maximale Zeit für RFC-Aufrufe in Sekunden. Beispiel:

RfcAccessTimeout 10

Nach 10 Sekunden wird der RFC-Aufruf abgebrochen.

`UserStatistics`

Angabe eines von allen Fronten-PCs beschreibbaren Verzeichnisses, in das GuiXT pro Benutzer folgende Informationen einträgt: SAP-System, Benutzername, SAP GUI Version, GuiXT Version, aktive GuiXT-Komponenten.

2.4. Ablage der Profile Parameter

Zur Ablage der GuiXT Profile-Parameter haben Sie folgende Möglichkeiten:

1. **Windows Registry**
2. **Lokale guixt.ini Datei**
3. **Zentrale guixt.ini Datei**
4. **Systemspezifische guixt.ini Datei**

Dabei geht GuiXT nach folgender Logik vor: Zunächst wird in dem laufenden Programmverzeichnis (s.u.) nach einer Datei **guixt.ini** gesucht. Wird diese gefunden, und ist dort die Zeile

IgnoreRegistry Yes

angegeben, so werden die Profile-Parameter aus der guixt.ini-Datei entnommen, sonst aus der Windows Registry. Durch eine Anweisung

SwitchTo . . . .

in der lokalen guixt.ini können Sie umschalten auf eine zentral abgelegte Datei. Diese kann auf einem File Server im Netzwerk liegen, auf einem http/ftp-Server oder im SAP Web Repository in der SAP-Datenbank. Im Fall des SAP Web Repository, oder wenn Sie in dem bei SwitchTo angegebenen Dateinamen die Variable **&database** verwenden, wird die guixt.ini Datei pro SAP System separat gelesen.

In welchem Verzeichnis wird guixt.ini gesucht?
Der Namen des Verzeichnisses hängt ab, wo sapgui bzw. ITS auf dem betreffenden Rechner installiert ist. Meist sind es die Verzeichnisse

C:\Programme\SAP\Frontend\SAPgui (sapgui 6.20)

C:\Programme\SAPpc\SAPgui (sapgui 4.6D)

C:\Programme\SAP\ITS\2.0\programs (ITS)

Sie können sich in sapgui über **Help->About->Loaded Dlls** anzeigen lassen, aus welchem Programmverzeichnis die SAP Programme und auch guixt.dll geladen wurden.

2.4.1. Windows Registry

Vorgehen: Keine guixt.ini Datei anlegen. Oder, falls guixt.ini existiert: dort n i c h t "IgnoreRegistry Yes" setzen

Die Ablage in der Windows Registry ist insbesondere während der Entwicklung von GuiXT-Skripten bequem, da dann alle Parameter über die GuiXT Profile-Pflege leicht variiert werden können.

Die Einträge in der Registry sind unter

HKEY_CURRENT_USER
Software\SAP\SAPGUI Front\SAP Frontend Server\Customize

zu finden. Eine direkte Bearbeitung (über regedit) statt über die GuiXT Profile-Pflege ist möglich, aber wie jede direkte Registry-Pflege etwas gefährlich.

Wenn Sie ein Tool einsetzen, das die Registry-Einträge auf allen Benutzer PCs zentral ändern kann, ist auch für eine grössere Benutzerzahl die Ablage über die Windows-Registry praktikabel. Ansonsten ist eine zentrale guixt.ini zu empfehlen.

2.4.2. Lokale guixt.ini Datei

Vorgehen: guixt.ini Datei anlegen, dort "IgnoreRegistry Yes" setzen

Die Namen der guixt.ini Parameter sind in der Profile Dokumentation jeweils angegeben. Sie können sie auch in der GuiXT Profilepflege direkt über die Help-Funktion ("?" rechts oben) anzeigen lassen. Ein zusätzlicher Tip: Der GuiXT Designer bietet die Möglichkeit, aus der Windows Registry eine guixt.ini Datei automatisch zu generieren.

2.4.3. Zentrale guixt.ini Datei

Vorgehen: guixt.ini Datei anlegen, dort "IgnoreRegistry Yes" setzen, mit "SwitchTo" auf zentrale Datei verweisen

Sie können von der lokalen guixt.ini Datei im sapgui-Verzeichnis umschalten auf eine zentrale guixt.ini Datei, um die die Profile-Parameter für viele Benutzer identisch zu halten. Geben Sie hierzu in der lokalen guixt.ini den Namen der zentralen Datei (z.B. \\server12\GuiXT\guixt.ini) unter "SwitchTo" an:

```
IgnoreRegistry Yes
SwitchTo  \\server12\GuiXT\guixt.ini
```

Falls die entsprechende Datei nicht gefunden wird, z.B. weil der Server nicht verfügbar ist, liest GuiXT in der lokalen .ini-Datei weiter. Dadurch haben Sie die Möglichkeit, mehrere SwitchTo-Zeilen untereinander zu schreiben, wobei nur die erste gefundene Datei ausgewertet wird:

```
IgnoreRegistry Yes
SwitchTo   \\server001\GuiXT\guixt.ini
SwitchTo   \\server002\GuiXT\guixt.ini
```

2.4.4. Systemspezifische guixt.ini Datei

Vorgehen: guixt.ini Datei anlegen, dort "IgnoreRegistry Yes" setzen, mit "SwitchTo" auf Datei "SAPWR:..." verweisen, oder &database im Dateinamen verwenden

Bei Verwendung des SAP Web Repository müssen Sie den RFC Benutzer mit Password bereits in der lokalen ini-Datei angeben, damit der Zugriff auf das WebRepository erfolgen kann. Ebenfalls bitte die GuiXT Startoptionen bereits in der lokalen guixt.ini angeben. Beispiel für eine entsprechende .ini-Datei:

```
IgnoreRegistry Yes
StartMinimized Yes
RfcUser        guixtrfc
RfcPasswordEnc vaqfxepfhfwflfdgag
SwitchTo       SAPWR:zguixt.ini.txt
```

Die Anmeldung zum Zugriff auf das Web Repository erfolgt im Mandant 000. Sie können einen abweichenden Mandanten bei Bedarf durch die Notation "SAPWR,client=xxx" mitgeben, z.B.:

```
IgnoreRegistry Yes
StartMinimized Yes
RfcUser        guixtrfc
RfcPasswordEnc vaqfxepfhfwflfdgag
SwitchTo       SAPWR,client=800:zguixt.ini.txt
```

Falls bei Anmeldung in einem SAP-System die angegebene systemspezifische guixt.ini-Datei nicht gefunden wird, oder der RFC-Benutzer dort nicht existiert, erfolgt keine Fehlermeldung, aber GuiXT ist in dem betreffenden System dann nicht aktiv. Sie finden im GuiXT-Window über View->Protocol einen Eintrag, dass die ini-Datei nicht gefunden wurde.

Ein Beispiel für eine systemabhängige ini-Datei unter Verwendung von "&database":

```
IgnoreRegistry Yes
StartMinimized Yes
SwitchTo       http://www.xxx.com/guixt/ini/guixt.&database.ini
```

Bei Anmeldung im System P46 wird dann die ini-Datei
`http://www.xxx.com/guixt/ini/guixt.P46.ini` gelesen (bitte Gross/Kleinschreibung auf UNIX-Servern beachten).

Schliesslich ist auch eine Kombination von 3 und 4 möglich, d.h. zunächst Verweis auf eine zentrale .ini-Datei, dort die RFC-Parameter setzen und auf eine systemspezifische ini-Datei verweisen.

2.5. Benutzerkatalog

Durch die Angabe der Skript-Verzeichnisse im GuiXT-Profile können auf unterschiedlichen PCs unterschiedliche GuiXT Skripte aktiviert werden. Dieses Verfahren ist manchmal nicht ausreichend:

- Falls Benutzer an wechselnden PCs arbeiten, soll die Einstellung der GuiXT Skripten nicht von dem PC, sondern von dem SAP-Benutzernamen abhängen
- Statt einer individuellen Einstellung auf jedem PC wird die zentralen Einstellung der Skriptverzeichnisse gewünscht

Beide Ziele können Sie durch die Angabe eines Benutzerkatalogs (Angabe "UserCatalog" in guixt.ini) erreichen. Gehen Sie dazu wie folgt vor:

1 Teilen Sie die Benutzer Ihres SAP Systems in Gruppen ein, für die unterschiedliche GuiXT Skripten gelten sollten. Hierzu erstellen Sie eine Textdatei (Name z.B. usercatalog.txt), in der die Benutzergruppen und die Zuordnung der Skriptverzeichnisse enthalten sind. Sie können entweder direkt den Benutzernamen oder Rollen und Profile verwenden.

Beispiel:

```
// Benutzergruppen:
// jeweils 1-4 Skriptverzeichnisse zuordnen

M1 \\P5008\GuiXT\General \\P5008\GuiXT\M1
M2 \\P5008\GuiXT\General \\P5008\GuiXT\M2
MX \\P5008\GuiXT\General
TEST \\P5008\Guixt\Test
XX none

// Benutzer:
// jeweils einer Benutzergruppe zuordnen

000/Adam M1
000/Rinaldini M2
000/Zwitschke M1
000/* MX // Sonstige Benutzer im Mandant 000

800/Anton M1
800/Bertini M1
800/Zencke TEST

// Alternativ (oder zusätzlich können Sie eine Rolle oder ein Profile abfragen durch
// Role=.... oder Profile=...
// und die entspreche3nden Benutzer eine Benutzergruppe zuordnen
// (gilt für alle Mandanten gleichzeitig)

Role=SD_FACTORY_SALES M1
Role=SAP_FI* M2
* XX // Sonstige Benutzer, alle Mandanten

// Die Abfrage auf den Wert eines Benutzerparameters, z.B. des Parameters ZZGXT auf "SALES", ist möglich
// durch die Notation Parameter.PID=Wert
// Beispiel:

Parameter.ZZGXT=SALES M1
Parameter.ZZGXT=FIN M2
```

Erläuterung des Formats:

- Mit // können Sie in den Skripten Kommentare einfügen, Leerzeilen werden ignoriert.
- Im ersten Teil des Benutzerkatalogs sind 5 Benutzergruppen benannt: M1, M2, MX, TEST, XX. Jeder Gruppe können bis zu 4 Skriptverzeichnisse zugeordnet werden. Bitte die Namen der Skriptverzeichnisse in "..." einschliessen, falls sie Leerzeichen enthalten.
- Im zweiten Teil des Benutzerkatalogs wird pro Mandant/Benutzername genau eine Benutzergruppe angegeben. Dabei können Sie am Ende jedes Mandanten die restlichen Benutzer durch die Mandant/*-Angabe ebenfalls zuordnen. Ebenso ist eine mandantenübergreifende *-Angabe am Ende des Katalogs möglich
- Eine alphabetische Sortierung der Benutzer oder Benutzergruppen ist nicht nötig, fördert aber sicher die Übersicht.
- Die Angabe **none** bewirkt, daß kein Verzeichnis aktiv ist. In diesem Fall werden auch eventuelle direkt angegebene Skriptverzeichnisse nicht wirksam

2 Geben Sie im GuiXT Profile den Benutzerkatalog an.

Zur zentralen Administration empfehlen wir eine zentrale .ini-Datei, d.h.

- Auf den einzelnen PCs liegt im SAP GUI-Verzeichnis immer die gleiche Datei guixt.ini, welche lediglich den Eintrag
  SwitchTo \\P5003\guixt\guixt.ini
  enthält, wobei P5003 ein zentraler Fileserver ist.
- Die Datei \\P5003\guixt\guixt.ini enthält die GuiXT-Optionen und verweist auf einen zentralen Benutzerkatalog, z.B.:

```
IgnoreRegistry Yes
StartHidden Yes
Component GuiXT
Component InputAssistant
Component Viewer
Key mycompany.C60.aghjll
Key mycompany.HR2.awertl
UserCatalog \\P5003\guixt\UserCatalog.txt
```

Fragen und Antworten zum Benutzerkatalog

Q:Wir haben zwei produktive Systeme: C60 und HR2. Kann pro System ein unterschiedlicher Benutzerkatalog angegeben werden?

A: Ja, das ist möglich über die Variable &database bei Angabe des UserCatalog, z.B.

```
UserCatalog \\P5003\GuiXT\&database\usercatalog.txt
```

Es wird dann in C60 der Benutzerkatalog \\P5003\GuiXT\C60\usercatalog.txt gelesen, in HR2 der Benutzerkatalog \\P5003\GuiXT\HR2\usercatalog.txt.

Q:Welches Skriptverzeichnis gilt für das Logon-Bild?

A: Der Benutzerkatalog wird erst nach dem Logon-Bild eingelesen. Für das Logonbild selbst gelten die direkt angegebenen Verzeichnisse.

Q:Was passiert, wenn ein Benutzer nicht in dem Benutzerkatalog gefunden wurde?

A: Falls eine passende *-Angabe existiert, wird die entsprechende Benutzergruppe zugeordnet. Andernfalls gelten die direkt angegebenen Verzeichnisse.

Q:Wirken sich Änderungen des Benutzerkatalogs sofort aus?

A: Nein, sie wirken pro Benutzer erst bei der nächsten neuen Anmeldung im SAP.

Q:Kann der Benutzerkatalog auch auf einem Webserver liegen?

A: Ja, geben Sie hierzu einfach die URL der Datei an, z.B.

UserCatalog http://www.mycompany.com/guixt/usercatalog.txt

2.6. GuiXT Script Ablage im Mime Repository

Allgemeines

Ab GuiXT Version 2011 Q1 1 können Sie GuiXT Skripte, Bilder (gif, jpg), InputScripte und beliebige andere Dateien auch im "SAP MIME Repository" ablegen, SAP Transaktion SE80. Gegenüber der älteren Ablage im SAP Web Repository bietet das einige Vorteile:

- SAP unterstützt die Versionierung aller dort abgelegten Dateien
- Die Dateien können in einer Baumstruktur übersichtlich angeordnet werden
- Insgesamt ist die Pflegetransaktion SE80 komfortabler als SMW0 (Web Repository). Sie wird von SAP auch für Web Dynpro und BSP-Anwendungen genutzt.
- Es steht ein Tool (ABAP-Programm BSP_UPDATE_MIMEREPOS) zur Verfügung, das lokale Verzeichnisse ins Mime Repository transportiert oder von dort in ein lokales Verzeichnis kopiert
- Der Zugriff kann auch über http oder WebDAV erfolgen. GuiXT greift über RFC-aufrufe auf die Inhalte des Mime-Repository zu.
- Die für das Web Repository nötige Freigabe des Zugriffs-Funktionsbausteins ist nicht nötig.

Die Nachteile:

- Der erste Zugriff ist etwas langsamer, da pro Datei insgesamt 3 RFC-Aufrufe nötig sind. Es empfiehlt sich, mit VersionNumber zu arbeiten, dann spielen die Zugriffszeiten im produktiven Einsatz praktisch keine Rolle.
- Durch die vielfältigen Möglichkeiten des Mime Repository ist alles etwas komplexer

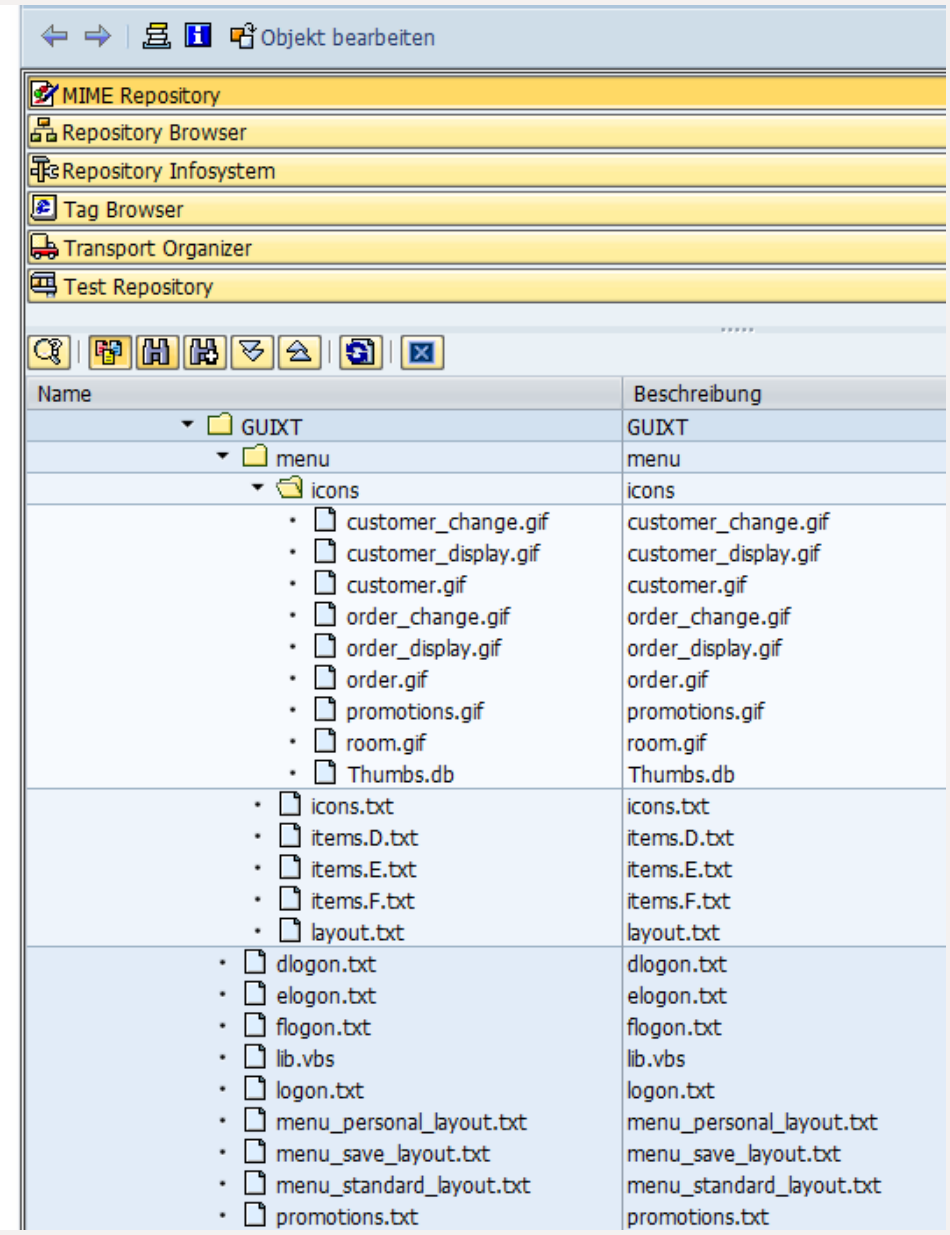
Durch die Notation **SAPMR:** analog zu SAPWR: können Sie überall, wo Dateien oder Verzeichnisse in GuiXT

eine Rolle spielen, einen Pfad im SAP Mime Repository angeben. Beispiele:

// GuiXT Profile
Directory1 SAPMR:SAP/BC/BSP/SAP/GUIXT

//Script
Image (toolbar) SAPMR:SAP/BC/BSP/SAP/GUIXT/ICONS/help.gif

Beispiel aus Transaktion SE80, SAP Mime Repository:



2.7. RFC (Remote Function Call) in GuiXT

GuiXT nutzt für verschiedene Funktionen eine RFC-Verbindung (Remote Function Call) zum SAP System. Zum Anmelden im System und für die Berechtigungsprüfung ist im RFC ein Benutzerstammsatz nötig. Welchen Benutzerstammsatz GuiXT verwendet, ist im GuiXT Profile hinterlegt. Als Default-Wert wird SAPCPIC vorgeschlagen; dieser Benutzerstammsatz ist in produktiven Systemen aber häufig gesperrt. Empfehlenswert ist, einen eigenen RFC-Benutzer für die GuiXT-RFC-Aufrufe anzulegen. Zu Testzwecken kann man auch den eigenen Benutzer (+ Passwort) im Profile eintragen.

In der folgenden Tabelle sind die einzelnen GuiXT-Aktionen, der benutzte Funktionsbaustein, der Mandant und die erforderlichen Berechtigungen des RFC-Benutzers aufgeführt. Statt des Mandanten 000 kann auch über eine Option im Profile bestimmt werden, dass GuiXT immer den aktuellen Mandant verwendet. Das hat den Vorteil, dass kein Benutzerstammsatz im Mandanten 000 angelegt werden muss, bei Verwendung mehrerer Mandanten allerdings den Nachteil, dass der Benutzer in jedem Mandanten existieren muss.

Das Anlegen des Benutzerstammsatzes und der entsprechenden Berechtigungen kann im SAP System über die Transaktionen **SU01, SU02, SU03** erfolgen. Die Vergabe z.B. der Berechtigungen für den Web Repository Zugriff ist unten dargestellt.

Aktion	Mandant / Funktionsbaustein	Erforderliche Berechtigungen des RFC Benutzers
Lesen Skripte in Transaktionsvarianten	000 RS_HDSYS_GET_GUIXT_SCRIPT	**S_RFC:** RFC_TYPE=FUGR, RFC_NAME=SHD0, ACTVT=16
Lesen Web Repository Objekte "SAPWR:..."	000 WWW_DISPATCH_REQUEST	**S_RFC:** RFC_TYPE=FUGR, RFC_NAME=SURL, ACTVT=16
Lesen Data Dictionary (symbolische offsets in Variablen)	000 RPY_TABLE_READ	**S_RFC:** RFC_TYPE=FUGR, RFC_NAME=SIFD, ACTVT=16 **S_DEVELOP:** DEVCLASS=*,OBJTYPE=TABL,OBJNAME=*,ACTVT=03
if Q[Role=...] und if Q[Profile=...]	aktueller Mandant BAPI_USER_GET_DETAIL	**S_RFC:** RFC_TYPE=FUGR, RFC_NAME=SU_USER, ACTVT=16 **S_USER_GRP:** CLASS=*,ACTVT=03 **S_USER_SYS:** CLASS=*,ACTVT=90 (falls zentrale Benutzerverwaltung)
Lesen Bilder aus "SAPDB:..."	000 RS_HDSYS_GET_GUIXT_IMAGE	**S_RFC:** RFC_TYPE=FUGR, RFC_NAME=RSIMG, ACTVT=16
Lesen Bilder aus "SAPIM:..." (Business Document Server)	000 RS_HDSYS_GET_BDS_IMAGE	**S_RFC:** RFC_TYPE=FUGR, RFC_NAME=RSIMG, ACTVT=16 **S_BDS_DS:** CLASSNAME=cccc, CLASSTYP=OT, ACTVT=30 dabei ist cccc der in der Bilddatei angegebene Klassenname

- Transaktion ST01: Berechtigungsprüfungen während RFC
- Transaktion ST05: RFC-Aufrufe
- Programm sapinfo in Verzeichnis sapgui/rfcsdk/bin: Test der RFC-Parametrisierung

Anlegen Benutzerstammsatz und Berechtigungen

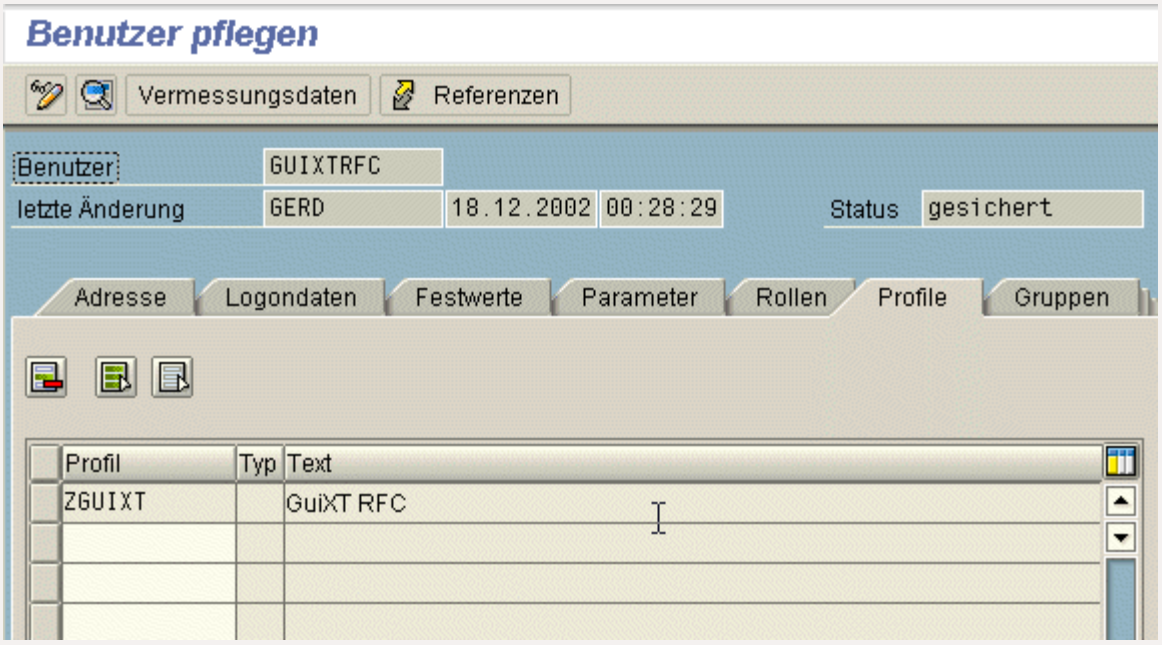


Abb. 1: Einstieg mit Transaktion SU01

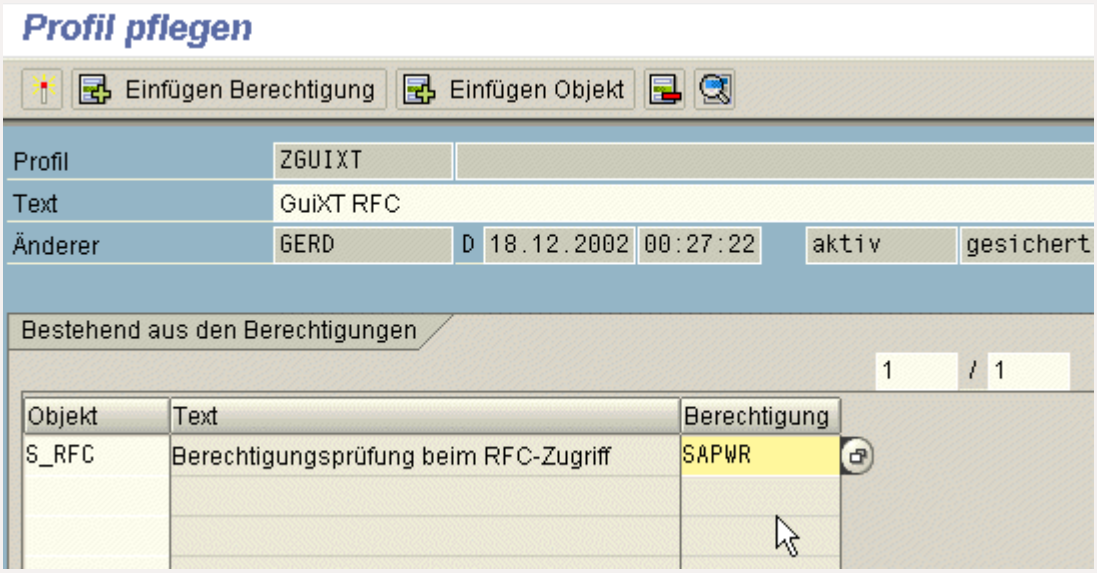


Abb. 2: Anlegen eines separaten Profils

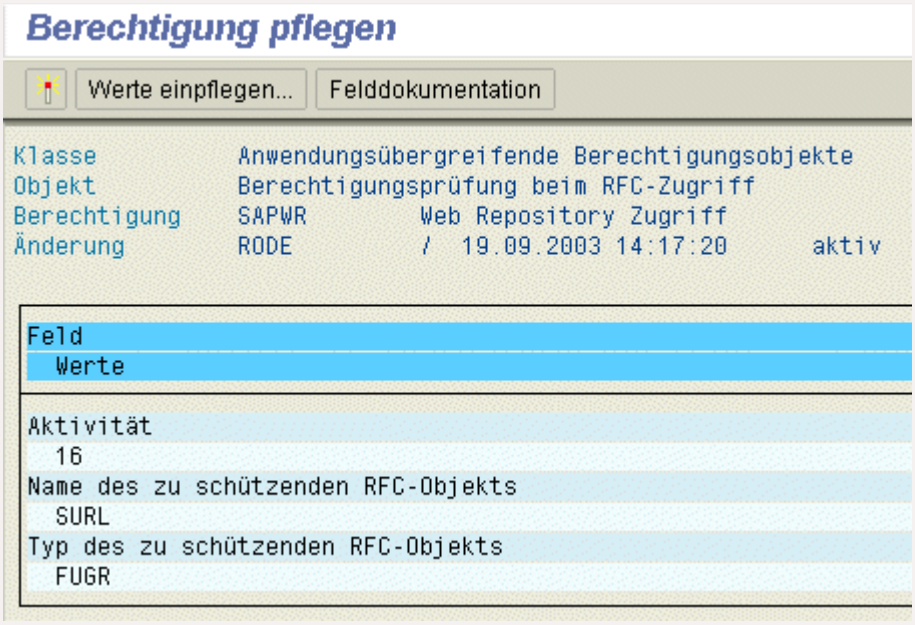


Abb. 3: Berechtigungen für Web Repository Zugriff laut Tabelle oben. Ebenso können die übrigen S_RFC Berechtigungen hier ergänzt werden.

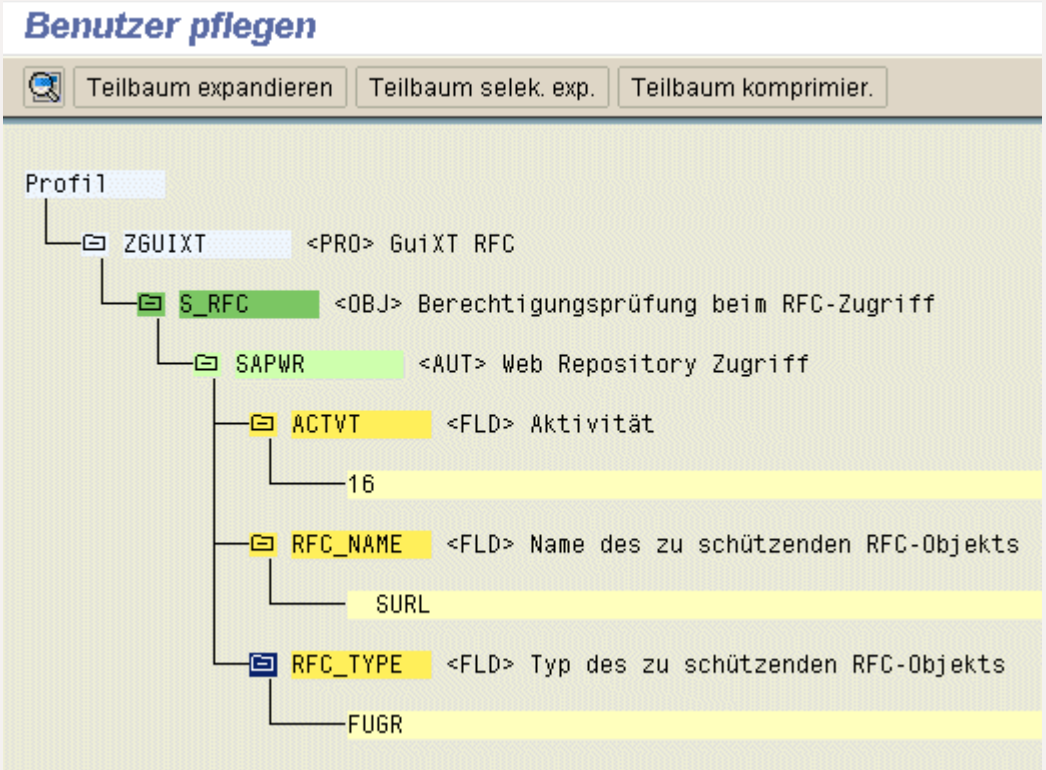


Abb. 4: Anzeige der angelegten Berechtigungen

2.8. Mehrsprachenfähigkeit

Sie haben zwei Möglichkeiten, Skripte für unterschiedliche Sprachen zu verwenden:

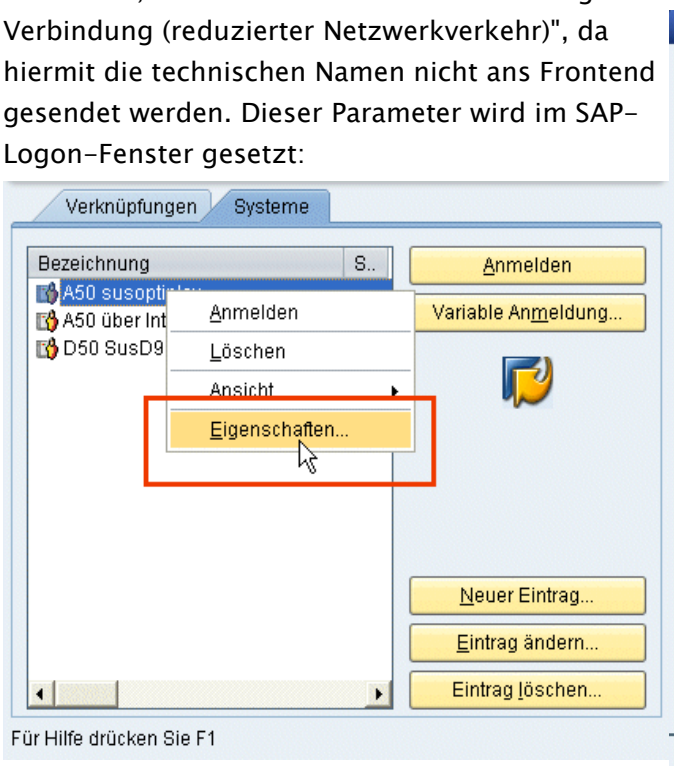
1. Sie verwenden ein jeweils unterschiedliches Set von Skripten für jede Sprache
2. Sie arbeiten mit einem einzigen Set von Skripten, wobei alle Sprachen innerhalb der Skripte abgehandelt werden.

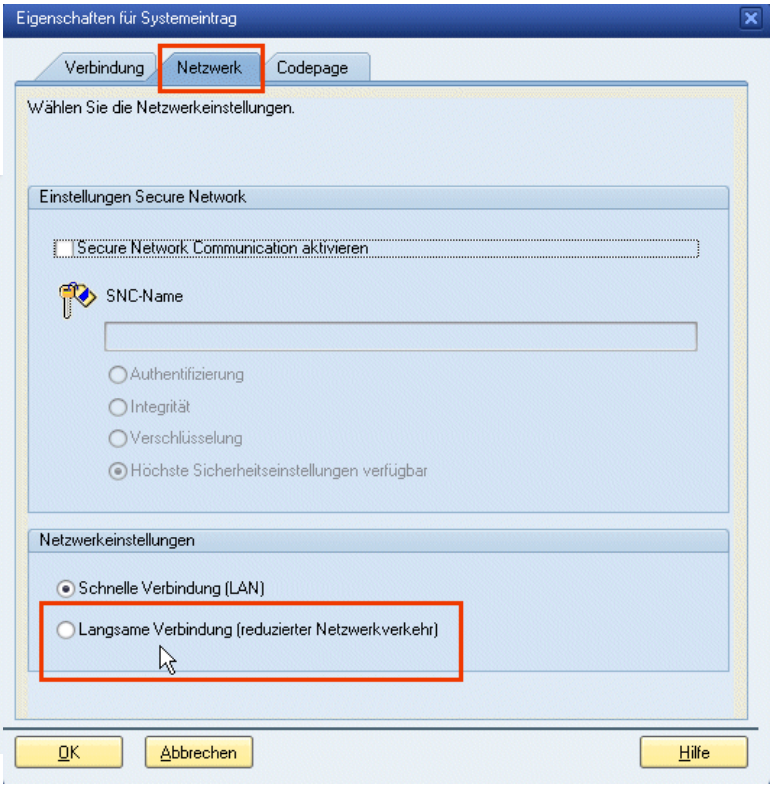
An dieser Stelle beschreiben wir nun die zweite Möglichkeit.

Ein Vorteil dieser zweiten Möglichkeit ist, dass die Wartung Ihrer Skripte vereinfacht wird: Sie unterteilen dazu jedes Skript in einen sprachenunabhängigen Teil, in dem Sie Felder löschen, neu anordnen, Defaults definieren usw. und einen sprachenspezifischen Teil, in dem Sie Felder umbenennen, Texte hinzufügen usw.

Ein Nachteil ist, daß Sie die technischen Namen der Felder in den sprachunabhängigem Teil einsetzen müßen.

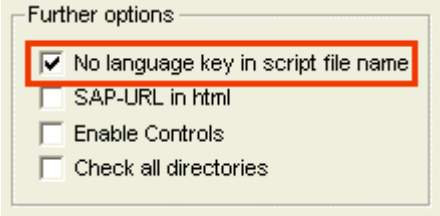
Außerdem, kann der Benutzer nicht mit "Langsame Verbindung (reduzierter Netzwerkverkehr)", da hiermit die technischen Namen nicht ans Frontend gesendet werden. Dieser Parameter wird im SAP-Logon-Fenster gesetzt:





Vorgehensweise:

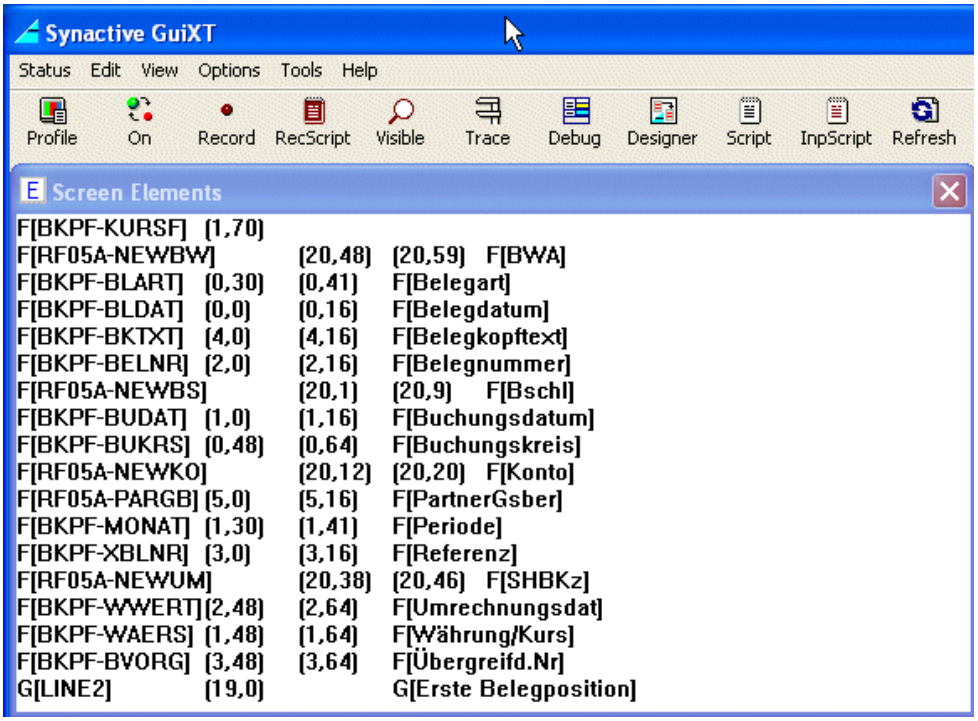
Im GuiXT Profile markieren Sie "No language key in script file name":



Wenn Sie mit einer guixt.ini Datei arbeiten, lautet der Parameter

NoLanguageKey Yes

GuiXT sucht infolgedessen nach Skripten ohne Sprachenschlüssel, z.B. anstelle der Skriptdatei SAPMF05A.D0100.txt liest es die Datei SAPMF05A.0100.txt. Zusätzlich finden Sie im GuiXT Screen Elements Fenster die technischen Feldnamen in der ersten Spalte:


Synactive GuiXT
Status Edit View Options Tools Help
Profile On Record RecScript Visible Trace Debug Designer Script InpScript Refresh
Screen Elements
F[BKPF-KURSF] (1,70)
F[RF05A-NEWBW] (20,48) (20,59) F[BWA]
F[BKPF-BLART] (0,30) (0,41) F[Belegart]
F[BKPF-BLDAT] (0,0) (0,16) F[Belegdatum]
F[BKPF-BKTXT] (4,0) (4,16) F[Belegkopftext]
F[BKPF-BELNR] (2,0) (2,16) F[Belegnummer]
F[RF05A-NEWBS] (20,1) (20,9) F[Bschl]
F[BKPF-BUDAT] (1,0) (1,16) F[Buchungsdatum]
F[BKPF-BUKRS] (0,48) (0,64) F[Buchungskreis]
F[RF05A-NEWKO] (20,12) (20,20) F[Konto]
F[RF05A-PARGB] (5,0) (5,16) F[PartnerGsber]
F[BKPF-MONAT] (1,30) (1,41) F[Periode]
F[BKPF-XBLNR] (3,0) (3,16) F[Referenz]
F[RF05A-NEWUM] (20,38) (20,46) F[SHBKz]
F[BKPF-WWERT] (2,48) (2,64) F[Umrechnungsdat]
F[BKPF-WAERS] (1,48) (1,64) F[Währung/Kurs]
F[BKPF-BVORG] (3,48) (3,64) F[Übergreifd.Nr]
G[LINE2] (19,0) G[Erste Belegposition]

Mit einem Mausklick auf das Element können Sie den technischen Namen in die Zwischenablage kopieren, um ihn dann mit Strg+V in Ihr Skript einzufügen.

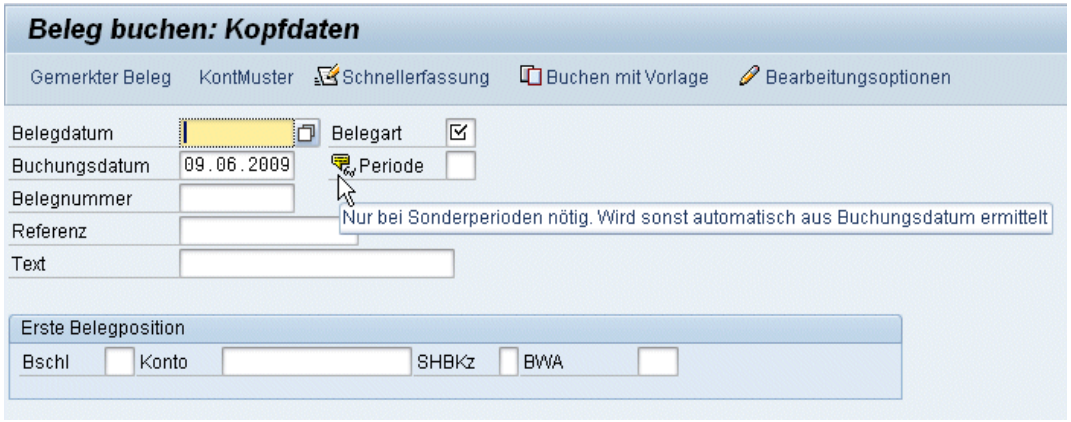
Für die Aufzeichnung eines InputScripts verwendet GuiXT ebenfalls die technischen Namen:

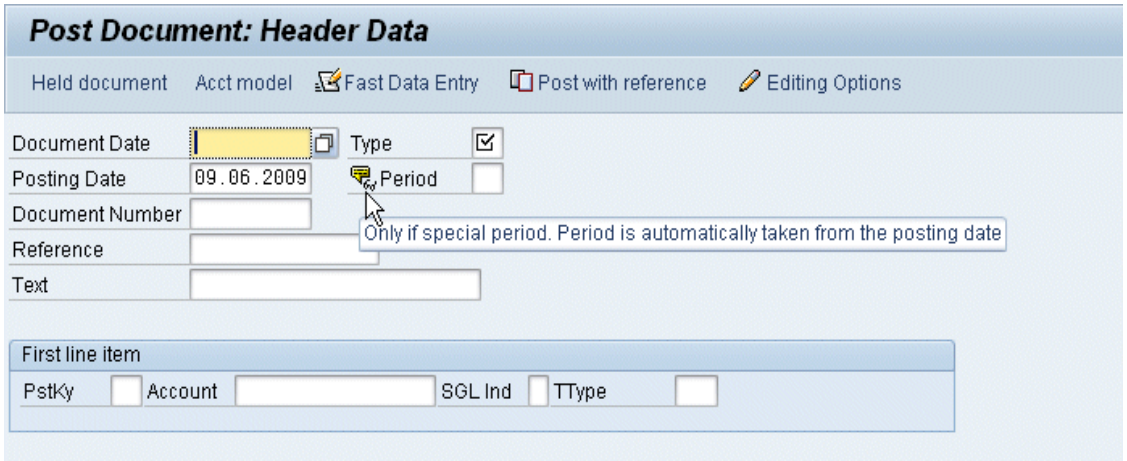
```
...
Screen SAPMF05A.0300
  SetCursor [RF05A-NEWKO]
  Set F[BSEG-WRBTR] "100"
  Set F[RF05A-NEWBS] "50"
  Set F[RF05A-NEWKO] "113101"
  Set F[COBL-GSBER] "0001"
  Enter

Screen SAPMF05A.0300
  SetCursor [BSEG-WRBTR]
  Set F[BSEG-WRBTR] "100"
  Enter "/11"
```

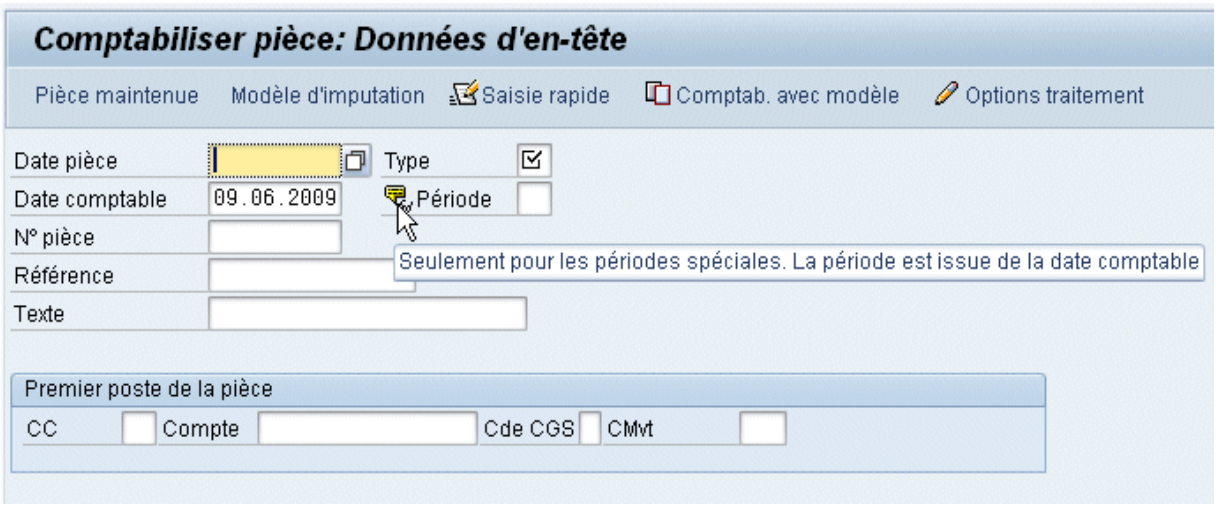
In Ihren Skripten können Sie mit einem sprachenunabhängigen Teil beginnen, in dem Sie alle Elemente mit Ihren technischen Namen bezeichnen. Beispiel:

```
// Sprachenunabhängiger Teil
del F[RF05A-PARGB]
del F[BKPF-WWERT]
del F[BKPF-BUKRS]
del F[BKPF-KURSF]
del F[BKPF-WAERS]
del F[BKPF-BVORG]

pos G[LINE2] (6,0)

Default F[BKPF-BUKRS] "0001"
Default F[BKPF-WAERS] "EUR"


// English
if Q[Language=E]
  Text [Doc. Header text] "Text"
  Tip [Period] "Only if special period. Period is automatically taken from the posting date"
endif

// Deutsch
if Q[Language=D]
  Text [Belegkopftext] "Text"
  Tip [Periode] "Nur bei Sonderperioden nötig. Wird sonst automatisch aus Buchungsdatum ermittelt"
endif


// Français
if Q[Language=F]
  Text [Texte d'en-tête] "Texte"
  Tip [Période] "Seulement pour les périodes spéciales. La période est issue de la date comptable"
endif
```

Hieraus ergeben sich folgende drei Bildschirme in den drei Sprachen:







In dem sprachenabhängigen Teil können Sie wahlweise die technischen Namen verwenden:

```
// English
if Q[Language=E]
  Text F[BKPF-BKTXT] "Text"
  Tip  F[BKPF-MONAT] "Only if special period. Period is automatically taken from the posting date"
endif

// Deutsch
if Q[Language=D]
  Text F[BKPF-BKTXT] "Text"
  Tip  F[BKPF-MONAT] "Nur bei Sonderperioden nötig. Wird sonst automatisch aus Buchungsdatum ermittelt"
endif


// Français
if Q[Language=F]
  Text F[BKPF-BKTXT] "Texte"
  Tip  F[BKPF-MONAT]"Seulement pour les périodes spéciales. La période est issue de la date comptable"
endif
```

2.9. Userstatistics

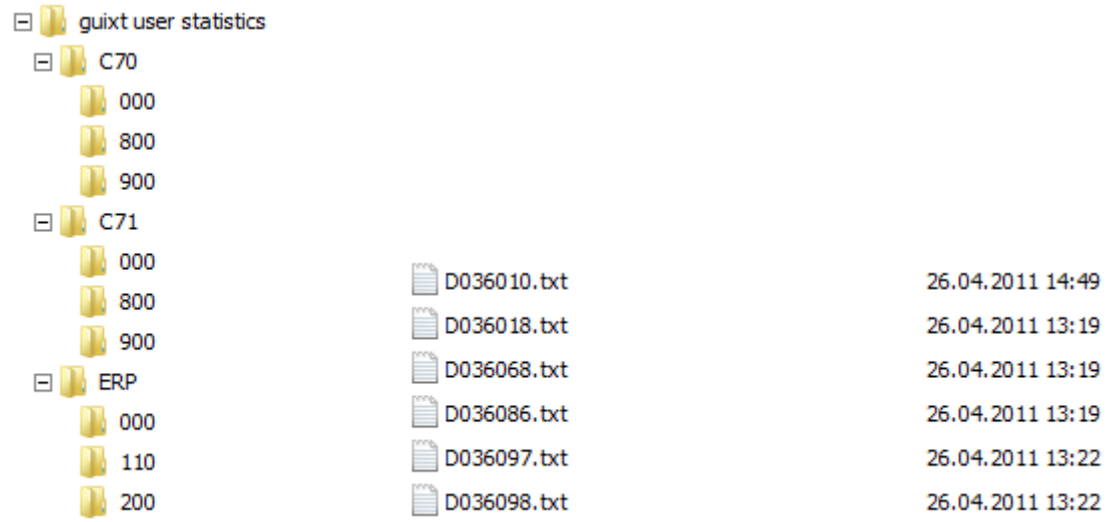
Durch Einschalten einer Statistikfunktion in der Konfigurationsdatei "guixt.ini" können Sie die Anzahl aktiver Benutzer sowie die jeweils verwendeten SAP GUI- und GuiXT-Versionen protokollieren. Voraussetzungen dazu sind:

- GuiXT ab Version 2011 Q1 4
- Ein zentrales Netzwerkverzeichnis mit Schreibberechtigung für alle Benutzer

In "guixt.ini" geben Sie das Verzeichnis, z.B. \\P712800\guixt\stats, wie folgt an:

```
UserStatistics \\P712800\guixt\stats
```

Bei jeder Neuanmeldung eines Benutzers in einem SAP-System wird dann bei aktiviertem GuiXT (SAP GUI-Optionen) eine Datei SYSTEMNAME\MANDANT\BENUTZERNAME.txt in dem angegebenen Verzeichnis gelesen und bei Bedarf geändert oder neu angelegt. Die Datei enthält die SAP GUI-Version, die GuiXT-Version sowie die aktiven GuiXT-Komponenten. Beispiel:




```
SAP GUI 710 PL10; GuiXT 2011 Q2 1;InputAssistant active;Viewer active;GuiXT Controls acti
```

Zur Auswertung der Informationen können Sie die Datei

guixtstat.hta

verwenden (HTA-Applikation, also im wesentlichen eine HTML-Datei).

Kopieren Sie die Datei in ein beliebiges Verzeichnis auf Ihrem PC. Sie können die Datei entweder direkt aufrufen (Doppelklick), in diesem Fall wählen Sie das Statistikverzeichnis in einem Benutzerdialog aus. Oder Sie geben das Verzeichnis als ersten Parameter beim Aufruf mit, am einfachsten in einer .bat-Datei:

```
guixtstat.hta "\\P712800\guixt\stats"
```

Es ist ratsam, die Statistik bei Bedarf in "guixt.ini" einzuschalten und dann einige Tage mitlaufen zu lassen, die Statistik dann wieder auszuschalten und das Statistikverzeichnis geeignet umzubenennen, sodass bei einem erneuten Einschalten wieder frische Daten erzeugt werden.

Die Auswertung durch "guixtstat.hta" zeigt folgende Informationen:

1. Anzahl GuiXT-Benutzer pro SAP-Systemname und Mandant, mit Summen
2. Verwendete GuiXT-Versionen
3. Verwendete SAP GUI-Versionen
4. Detailliste pro Benutzername

Bitte beachten Sie, dass die Gesamtzahl der Benutzer am Ende der Tabelle 1 höher sein kann als die Gesamtzahl am Ende der Detailliste. In unserem Beispiel unten z.B. 95 in Tabelle 1 und 80 am Ende der Detailliste. Das liegt daran, dass in der ersten Tabelle Benutzer mit gleichem Namen, die in unterschiedlichen Systemen oder Mandanten gearbeitet haben, jeweils separat gezählt werden, während die Detailliste pro Benutzernamen die jeweiligen Systeme und Mandanten aufführt, aber insgesamt jeden Namen nur einmal zählt.

Für die GuiXT-Lizenzen ist die Anzahl tatsächlicher Benutzer relevant. Falls über alle Systeme und Mandanten hinweg eindeutige Benutzernamen verwendet werden, ist das die Gesamtzahl am Ende der Detailliste.

GuiXT Statistics

Generated **26.04.2011** from **S:temp\guixt user statistics**

1. SAP systems and clients			
System	Client	Users	
C70	000	5	
	800	20	
	900	8	
		33	*
C71	000	3	
	800	14	
	900	8	
		25	*
ERP	000	5	
	110	24	
	200	8	
		37	*
Total		**95**	**

2. GuiXT versions	
GuiXT version	Users
GuiXT 2011 Q1 4	36
GuiXT 2011 Q1 5	29
GuiXT 2011 Q2 1	30

© by Synactive GmbH

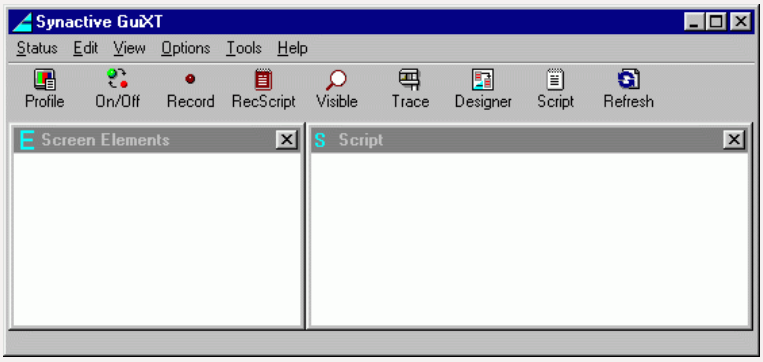
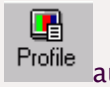
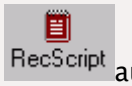
3. SAP GUI versions	
SAP GUI version	Users
SAP GUI 710 PL10	2
SAP GUI 710 PL2	1
SAP GUI 720 PL0	1
SAP GUI 720 PL2	40
SAP GUI 720 PL4	25
SAP GUI 720 PL5	3
SAP GUI 720 PL6	23

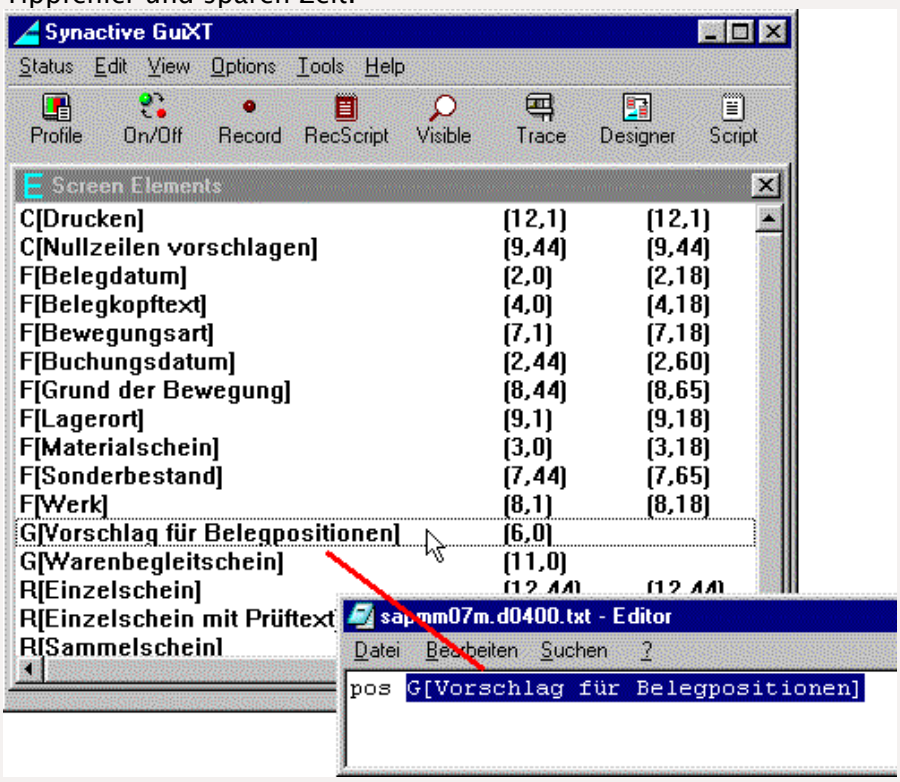
4. Users by name			
User	Systems/Clients	GuiXT Version	SAP GUI Version
D028325	C71/900	GuiXT 2011 Q2 1	SAP GUI 720 PL6
D031010	ERP/110	GuiXT 2011 Q1 5	SAP GUI 720 PL4
D031016	ERP/110	GuiXT 2011 Q1 5	SAP GUI 720 PL4
D031018	ERP/110	GuiXT 2011 Q1 5	SAP GUI 710 PL10
D031086	ERP/110	GuiXT 2011 Q1 4	SAP GUI 720 PL2
D031096e	ERP/110	GuiXT 2011 Q2 1	SAP GUI 720 PL6
D031097	ERP/110	GuiXT 2011 Q2 1	SAP GUI 720 PL6
D031111	ERP/110	GuiXT 2011 Q1 5	SAP GUI 720 PL4
D031125	ERP/110	GuiXT 2011 Q2 1	SAP GUI 720 PL6
D031128	ERP/110	GuiXT 2011 Q2 1	SAP GUI 710 PL2
D031139	ERP/110	GuiXT 2011 Q1 4	SAP GUI 720 PL2
D031145	ERP/110	GuiXT 2011 Q2 1	SAP GUI 720 PL6
D031169	ERP/110	GuiXT 2011 Q1 4	SAP GUI 720 PL2
D031183	ERP/110	GuiXT 2011 Q1 4	SAP GUI 720 PL2
D031184	ERP/110	GuiXT 2011 Q1 4	SAP GUI 720 PL2
D031211	ERP/110	GuiXT 2011 Q1 5	SAP GUI 720 PL4
D031212	ERP/110	GuiXT 2011 Q1 4	SAP GUI 720 PL2
D031216	ERP/110	GuiXT 2011 Q1 4	SAP GUI 720 PL2
D031220	ERP/110	GuiXT 2011 Q1 5	SAP GUI 720 PL4
D031225	ERP/110	GuiXT 2011 Q1 4	SAP GUI 720 PL2
D031229	ERP/110	GuiXT 2011 Q1 5	SAP GUI 720 PL4
D031232	ERP/110	GuiXT 2011 Q1 5	SAP GUI 720 PL2
D031239	ERP/110	GuiXT 2011 Q1 5	SAP GUI 720 PL2
D031281	ERP/110	GuiXT 2011 Q1 4	SAP GUI 720 PL2
D031283	ERP/110	GuiXT 2011 Q1 4	SAP GUI 720 PL2

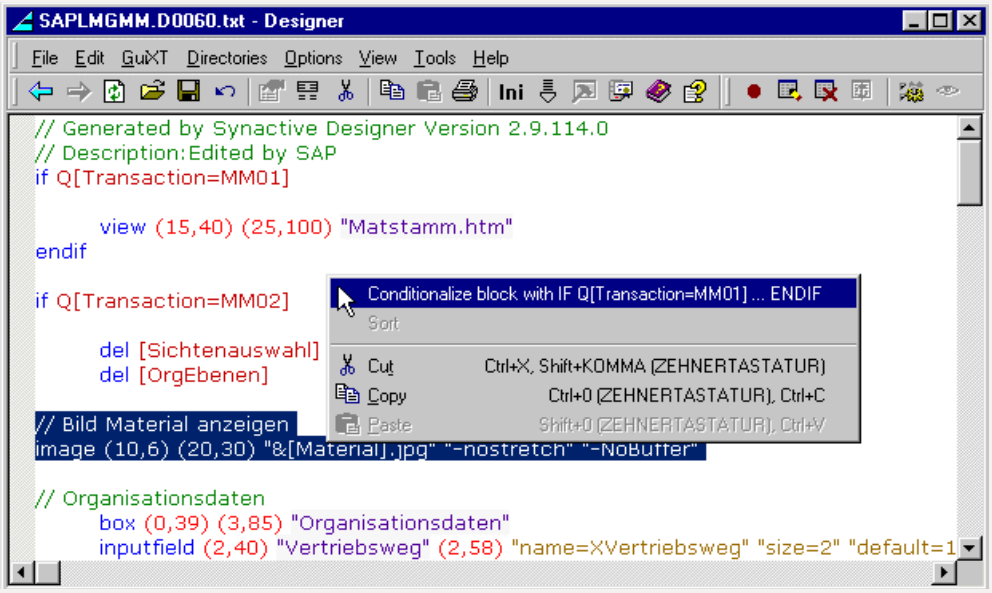
D038158	C71/800	GuiXT 2011 Q1 4	SAP GUI 720 PL2
D038194	C71/800	GuiXT 2011 Q1 4	SAP GUI 720 PL2
D038204	C71/800	GuiXT 2011 Q2 1	SAP GUI 720 PL4
D038225	C71/800	GuiXT 2011 Q2 1	SAP GUI 720 PL6
D038281	C71/800	GuiXT 2011 Q1 4	SAP GUI 720 PL2
D038282	C71/800	GuiXT 2011 Q1 4	SAP GUI 720 PL2
D038284	C71/800	GuiXT 2011 Q1 4	SAP GUI 720 PL2
D038297	C70/900 ERP/200	GuiXT 2011 Q2 1 GuiXT 2011 Q2 1	SAP GUI 720 PL6 SAP GUI 720 PL6
D038325	C70/900 ERP/200	GuiXT 2011 Q2 1 GuiXT 2011 Q2 1	SAP GUI 720 PL6 SAP GUI 720 PL6
D038486	C70/900 ERP/200	GuiXT 2011 Q1 4 GuiXT 2011 Q1 4	SAP GUI 720 PL2 SAP GUI 720 PL2
S1012	C70/000 ERP/000	GuiXT 2011 Q2 1 GuiXT 2011 Q2 1	SAP GUI 720 PL6 SAP GUI 720 PL6
S1016	C71/000	GuiXT 2011 Q2 1	SAP GUI 720 PL6
S1024	C70/000 ERP/000	GuiXT 2011 Q2 1 GuiXT 2011 Q2 1	SAP GUI 720 PL5 SAP GUI 720 PL5
S1044	C70/000 C71/000 ERP/000	GuiXT 2011 Q2 1 GuiXT 2011 Q2 1 GuiXT 2011 Q2 1	SAP GUI 720 PL6 SAP GUI 720 PL6 SAP GUI 720 PL6
S1046	C71/000	GuiXT 2011 Q2 1	SAP GUI 720 PL5
S1051	C70/000 ERP/000	GuiXT 2011 Q1 4 GuiXT 2011 Q1 4	SAP GUI 720 PL2 SAP GUI 720 PL2
S1083	C70/000 ERP/000	GuiXT 2011 Q2 1 GuiXT 2011 Q2 1	SAP GUI 720 PL6 SAP GUI 720 PL6
Total number of distinct user names: 80			

3. GuiXT Bedienung

3.1. Das GuiXT Fenster und Editor

Das GuiXT Fenster	Zentrale Schaltstelle für GuiXT und alle Komponenten ist das GuiXT Fenster. Standardmäßig öffnen sich beim Start die Anzeigefenster für Screen Elements und Script. 
Ansichtsoptionen	Im View Menü können Sie bestimmen, welche weiteren Anzeigefenster geöffnet werden.
Startoptionen	Beim Start von GuiXT kann das GuiXT Fenster sichtbar ("visible"), minimiert und sichtbar ("minimized") oder versteckt ("hidden") sein. Bitte beachten Sie hierzu die entsprechende Konfiguration des Start Window im GuiXT Profile, das Sie über den Button  aufrufen können.
Der Editor	GuiXT verwendet standardmäßig das Programm MS **Notepad** als Skripteditor. Der Editor wird im GuiXT Fenster durch die Befehle Edit --> Script, Edit --> History und Edit --> Recorded InputScript oder die Buttons und  aufgerufen. Der Name des Skripts wird automatisch aus dem jeweils aktuellen SAP Bild

	generiert. Siehe "Unter welchem Namen sind die GuiXT Skripte abgelegt?"
Alternativer Editor	Sie können wahlweise einen alternativen Editor benennen. Rufen Sie dazu mit dem Button das GuiXT Profile auf und wählen Sie das entsprechende Editorprogramm Ihrer Wahl aus.
Tipps &Tricks	Feldnamen können Sie einfach im GuiXT Screen Elements Fenster durch Doppelklick auf die entsprechende Feldbezeichnung in die Zwischenablage kopieren und dann mit Einfügen in Ihr Skript übernehmen. Sie vermeiden dadurch Tippfehler und sparen Zeit.  **Einfaches Einfügen von Feldnamen aus dem Screen Elements Fenster in den Editor** Sie können auch den integrierten Texteditor des Designer nutzen. Dieser zeigt Ihnen die Struktur des Skripts farblich aufbereitet an und stellt Ihnen Editierhilfsmittel zur Verfügung. Aufruf über den Designer, View -> Wysiwyg



Der integrierte Editor des Designer

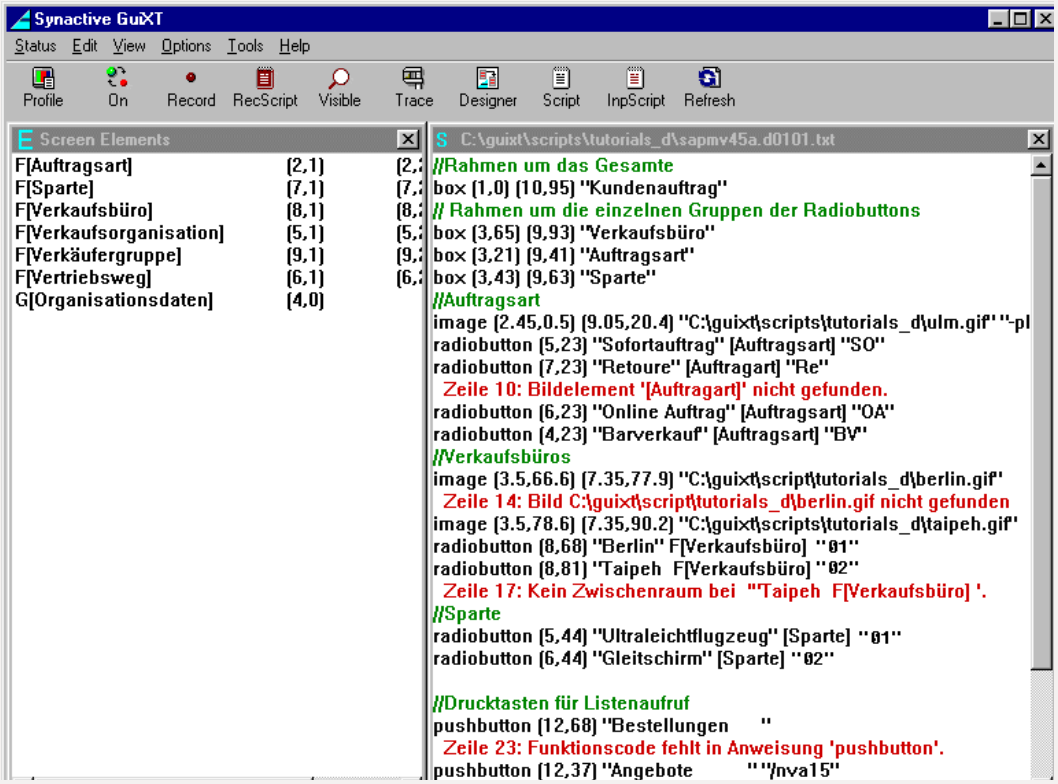
3.2. Fehlersuche

Vorweg:

Häufige Fehler in GuiXT Skripten sind einfache Tippfehler. Sie vermeiden diese, wenn Sie beim Erstellen der Skripte die Bildelemente aus dem Screen Elements Fenster durch Doppelklick in die Zwischenablage kopieren und dann in ihr Skript einfügen. Vgl. Editor/Tipps und Tricks

1. Überprüfen Sie in aller Ruhe, ob sich nicht in den von Ihnen geschriebenen Teilen (z.B. dem GuiXT Profile oder einem GuiXT Skript) ein kleiner Fehler eingeschlichen hat.

Fehler in GuiXT Skripts werden im GuiXT Fenster zur Laufzeit angezeigt. (**View->Script**) Sie erhalten einen kurzen Hinweis, worin der Fehler liegen könnte, z.B. fehlende Anführungszeichen, fehlende Koordinaten etc.


Synactive GuiXT
Status Edit View Options Tools Help
Profile On Record RecScript Visible Trace Designer Script InpScript Refresh
Screen Elements
F[Auftragsart] (2,1)
F[Sparte] (7,1)
F[Verkaufsbüro] (8,1)
F[Verkaufsorganisation] (5,1)
F[Verkäufergruppe] (9,1)
F[Vertriebsweg] (6,1)
G[Organisationsdaten] (4,0)
C:\guixt\scripts\tutorials_d\sapmv45a.d0101.txt
//Rahmen um das Gesamte
box (1,0) (10,95) "Kundenauftrag"
// Rahmen um die einzelnen Gruppen der Radiobuttons
box (3,65) (9,93) "Verkaufsbüro"
box (3,21) (9,41) "Auftragsart"
box (3,43) (9,63) "Sparte"
//Auftragsart
image (2.45,0.5) (9.05,20.4) "C:\guixt\scripts\tutorials_d\ulm.gif" "-pl
radiobutton (5,23) "Sofortauftrag" [Auftragsart] "SO"
radiobutton (7,23) "Retoure" [Auftragart] "Re"
Zeile 10: Bildelement '[Auftragart]' nicht gefunden.
radiobutton (6,23) "Online Auftrag" [Auftragsart] "OA"
radiobutton (4,23) "Barverkauf" [Auftragsart] "BV"
//Verkaufsbüros
image (3.5,66.6) (7.35,77.9) "C:\guixt\script\tutorials_d\berlin.gif"
Zeile 14: Bild C:\guixt\script\tutorials_d\berlin.gif nicht gefunden
image (3.5,78.6) (7.35,90.2) "C:\guixt\scripts\tutorials_d\taipeh.gif"
radiobutton (8,68) "Berlin" F[Verkaufsbüro] "01"
radiobutton (8,81) "Taipeh F[Verkaufsbüro] "02"
Zeile 17: Kein Zwischenraum bei "Taipeh F[Verkaufsbüro] '.
//Sparte
radiobutton (5,44) "Ultraleichtflugzeug" [Sparte] "01"
radiobutton (6,44) "Gleitschirm" [Sparte] "02"
//Drucktasten für Listenaufruf
pushbutton (12,68) "Bestellungen "
Zeile 23: Funktionscode fehlt in Anweisung 'pushbutton'.
pushbutton (12,37) "Angebote "/nva15"

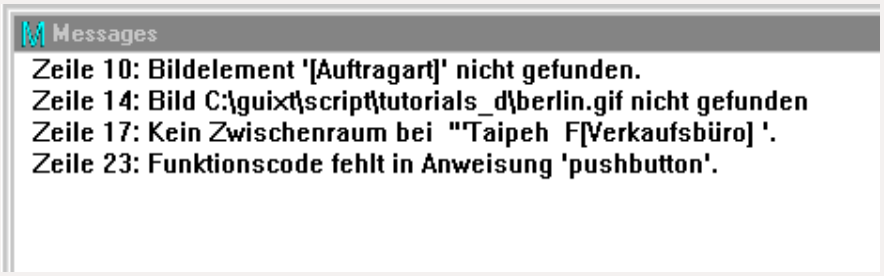
Einfache Fehlermeldungen werden im Script Fenster in Rot angezeigt

2. Wenn Sie der Ansicht sind, daß in Ihren Teilen alles stimmt:

Bitten Sie zur Sicherheit eine Kollegin/einen Kollegen, darüberzuschauen. Manchmal übersieht man selbst bei 3-maligem Lesen einen kleinen unscheinbaren Schreibfehler.

3. Nutzen Sie alle Hilfsmittel, die Sie auf eigene Fehler aufmerksam machen können:

Schauen Sie sich insbesondere das Message-Protokoll bei GuiXT an (**View->Messages**). Dies ist besonders bei längeren Skripten sinnvoll.


Messages
Zeile 10: Bildelement '[Auftragart]' nicht gefunden.
Zeile 14: Bild C:\guixt\script\tutorials_d\berlin.gif nicht gefunden
Zeile 17: Kein Zwischenraum bei "Taipeh F[Verkaufsbüro] '.
Zeile 23: Funktionscode fehlt in Anweisung 'pushbutton'.

	Im Message Protokoll sind die Fehlermeldungen übersichtlich nach Skriptzeilen geordnet.
4. Handelt es sich um etwas, was "früher" einmal lief, jetzt aber ohne irgendwelche Änderungen Ihrerseits nicht mehr funktioniert?	Häufig wurde in einem solchen Fall etwas im Umfeld geändert, ohne daß man sich aller Auswirkungen bewußt war; es wurde z.B. ein Directory umbenannt. Versuchen Sie, die Änderungen im Umfeld des Produkts temporär zurückzunehmen und mit dem alten Stand zu testen.
5. Ziehen Sie die Dokumentation zu Rate.	Insbesondere die Rubrik "Tips&Tricks" zu den einzelnen Anweisungen macht Sie vielleicht auf eine Besonderheit aufmerksam, die des Rätsels Lösung ist.
6. Konsultieren Sie unsere Q&A-Seiten.	Vielleicht ist dort bereits Ihr Problem und eine Lösung beschrieben. Bekannte offene Probleme und Korrekturen für ausgelieferte Versionen finden Sie unter Download - Version Info.
7. Support durch Synactive GmbH	Wenn Sie den Fehler nicht selbst finden oder beheben können, senden Sie uns bitte eine Problemmeldung. Diese bearbeiten wir gerne im Rahmen eines Wartungsvertrages (Support).

3.3. System Trace

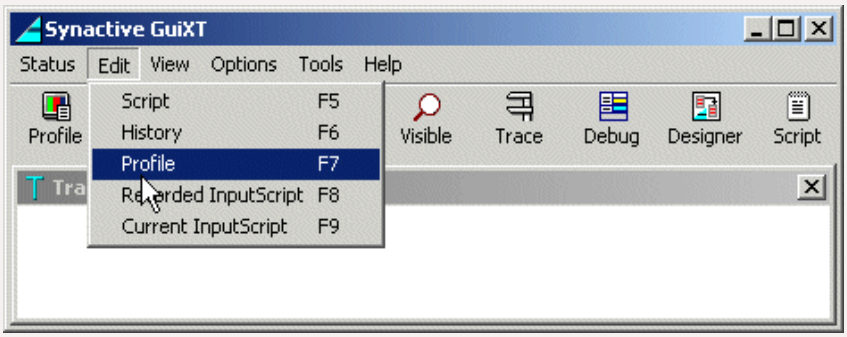
Zur Analyse von Problemen bei GuiXT Scripten ist es manchmal nötig, einen "GuiXT System Trace" zu erzeugen und an Synactive zu schicken. Dieser Trace enthält in der Regel alle Informationen, die zum Nachvollziehen des Problems gebraucht werden, was sonst manchmal schwierig ist, da der Releasestand des SAP-Systems, die Anwendungskonfiguration und die Anwendungsdaten eine Rolle spielen können.

Der Ablauf im Prinzip: Einschalten des System-Trace, Durchführen der entsprechenden Anwendungsschritte, Ausschalten des Trace, Speichern und Versand der Trace-Datei, eventuell als .zip Datei falls die Datenmenge relativ gross ist.

Falls Sie eine guixt.ini-Datei verwenden, lesen Sie bitte hier weiter.

1 Aktivieren Sie die Trace-Option

Wählen Sie **Edit->Profile** im GuiXT-Fenster oder drücken Sie auf das Symbol



Klicken Sie auf "**On**" in dem "**System Trace**" Kästchen

Sie können in dem Eingabefeld einen Dateinamen eintragen, z.B. C:\temp\guixt.trace.txt. Der Trace wird dann unmittelbar in diese Datei geschrieben. Wenn Sie das Eingabefeld für den Dateinamen leer lassen, wird der Trace nur temporär erzeugt und Sie müssen ihn mit Tools->Download Trace später in eine Datei sichern (siehe unten).

Klicken Sie auf "**OK**" im Profile-Fenster.

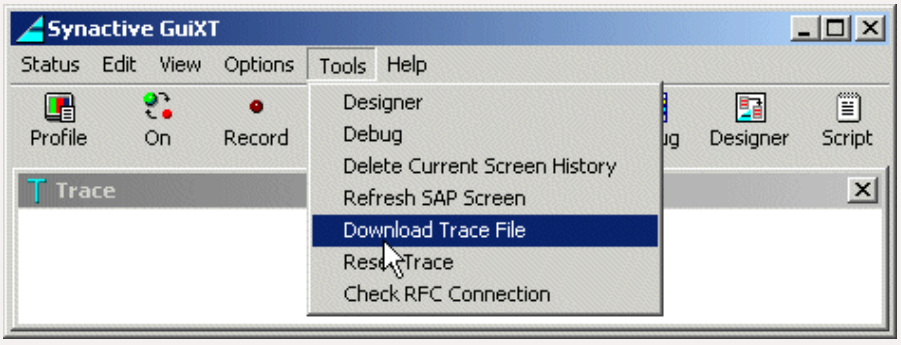
2 Reproduzieren Sie den Fehler

Führen Sie die SAP-Transaktion durch, für die das GuiXT-Skript nicht ordentlich funktioniert

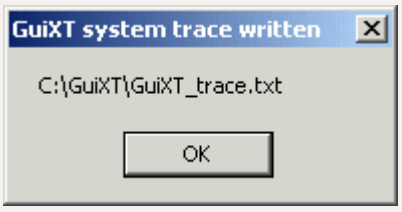
3 Speichern Sie die Trace-Datei

Klicken Sie auf **Tools->Download Trace File**

Nur nötig falls Sie keinen Dateinamen beim Einschalten angegeben haben

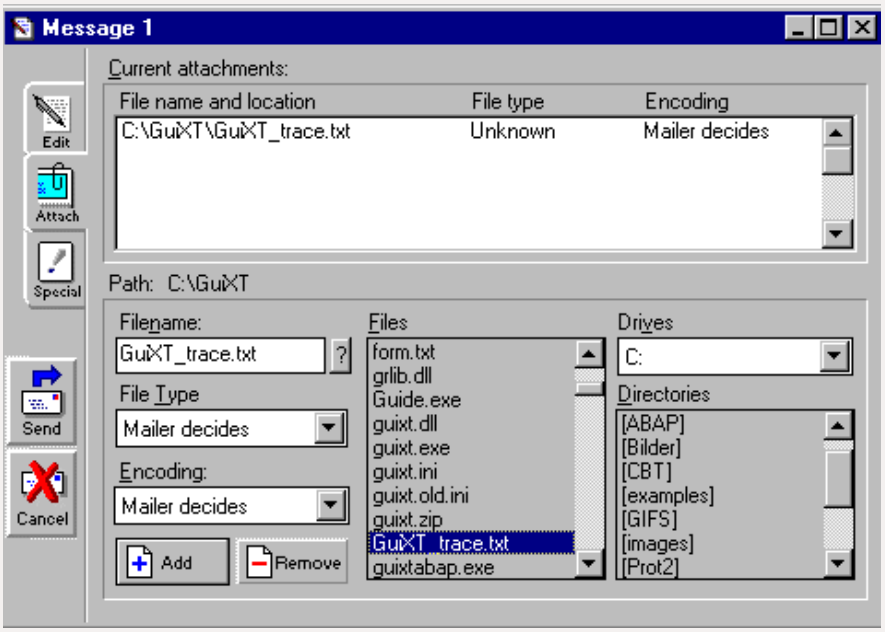


Sie erhalten folgende Popup-Nachricht

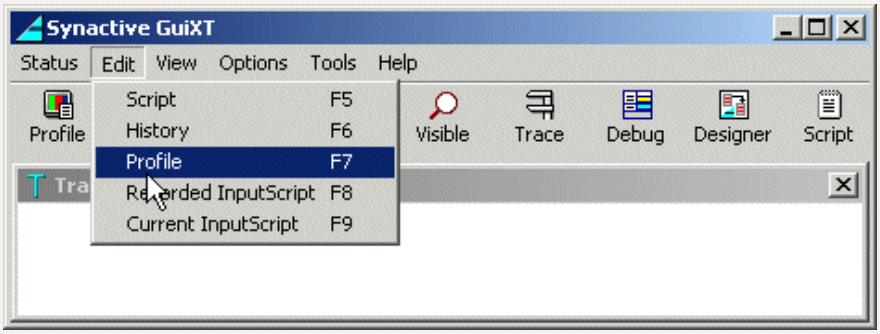


4 Senden Sie uns die Trace-Datei via email

Fügen Sie die Datei Ihrer email zu und senden Sie sie an support@synactive.com



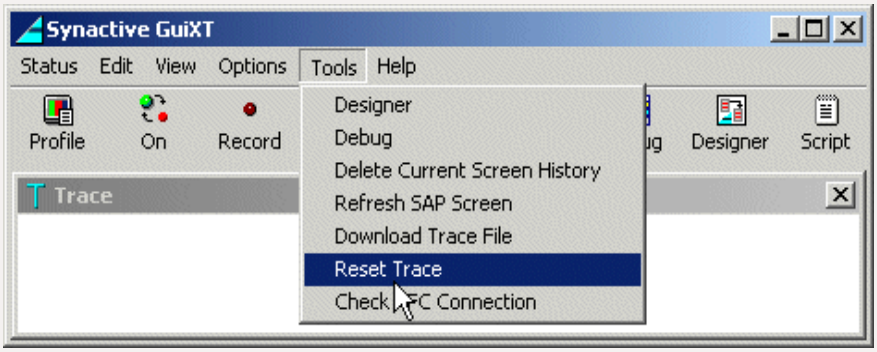
5 Deaktivieren Sie die Trace-Erstellung



Klicken Sie auf "Off" im "System Trace" Kästchen

Klicken Sie auf "OK" im Profile-Fenster.

Tipp: Zurücksetzen der Trace-Datei



Falls Sie Fehler bei der Erstellung des Trace gemacht haben, können Sie die Datei zurückzusetzen indem Sie auf "Reset Trace" klicken und das Trace erneut zu erstellen.

Erzeugen des Trace bei Verwendung einer guixt.ini Datei

Wenn Sie als GuiXT Profile eine .ini-Datei verwenden, ist das Vorgehen wie oben beschrieben nicht möglich. Eventuell ist auch das GuiXT Fenster nicht sichtbar, da GuiXT "hidden" gestartet wird.

Das Einschalten des Systemtrace ist in der guixt.ini-Datei möglich (Parameter "Trace" und "TraceFile"), aber oft möchten Sie die guixt.ini Datei nicht ändern, da sie zentral abgelegt ist und auch für andere Benutzer gilt. Ausserdem wird durch das Einschalten in guixt.ini immer der gesamte Ablauf, von login bis logout, protokolliert.

Aus diesen Gründen ist es empfehlenswert, den Trace durch Konsolenkommandos direkt ein- und auszuschalten. Das geht wie folgt:

Dynamisches Einschalten des Systemtrace

Geben Sie in Windows folgendes Konsolenkommando ein:

guixt.exe -traceon file=C:\temp\guixttrace.txt

Dabei den gewünschten Dateinamen setzen

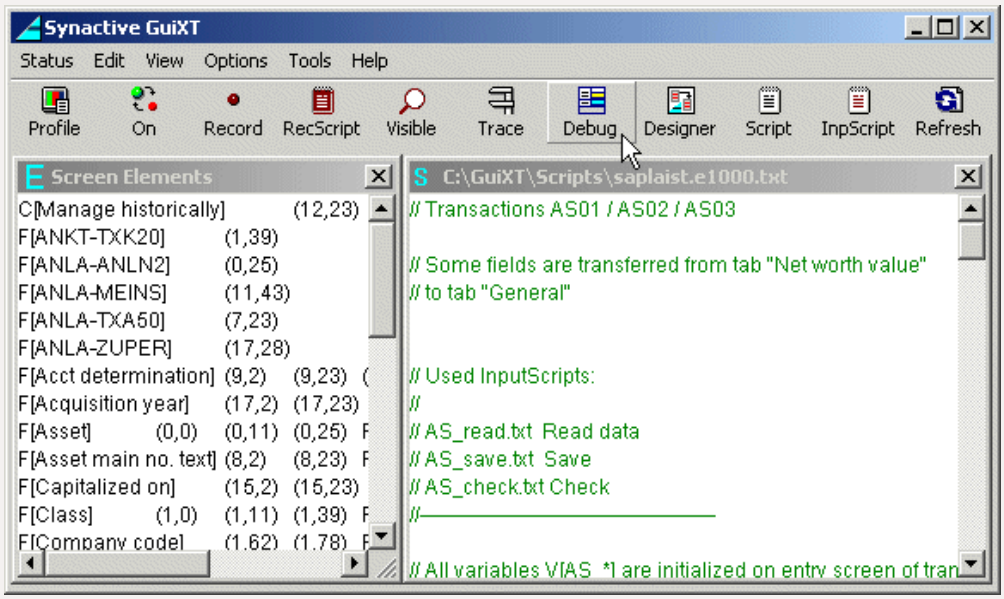
Dynamisches Ausschalten des Systemtrace

Geben Sie in Windows folgendes Konsolenkommando ein:

guixt.exe -traceoff

3.4. Debugger

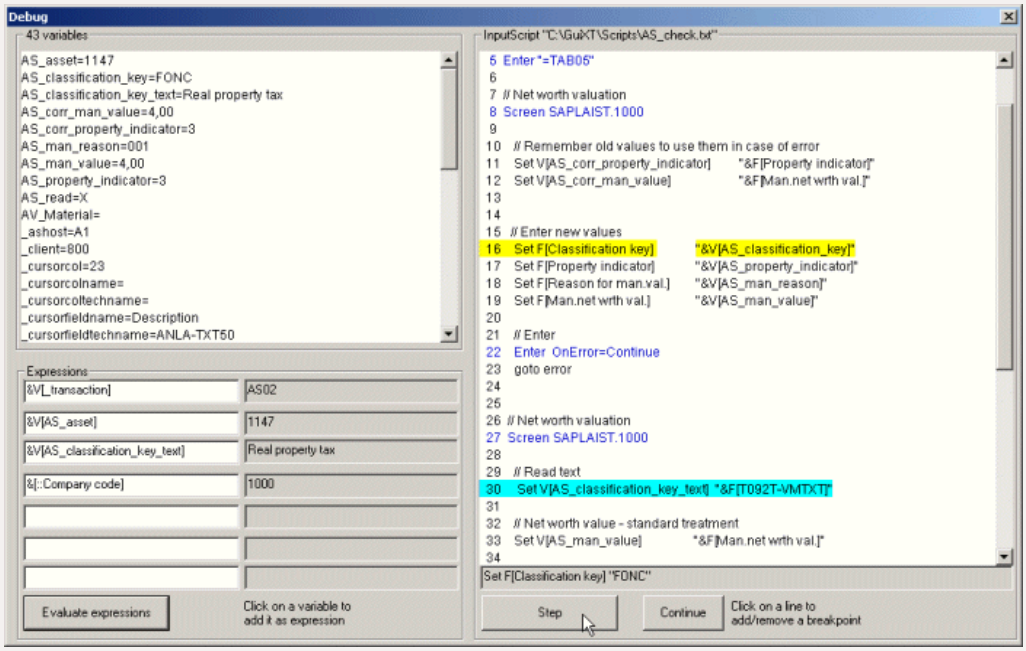
Der GuiXT Debugger wird durch Drücken auf das "Debug" Icon im GuiXT-Fenster gestartet:



Ein "Debug"-Fenster wird angezeigt (siehe Bild unten). Sobald ein InputScript ausgeführt wird und das Debug-Fenster offen ist, wird die Ausführung des InputScript automatisch in den Debug-Modus geschaltet. Wenn Sie das Debug-Fenster während der Ausführung des InputScript schliessen, wird der Rest im normalem Modus (nicht debug) ausgeführt.

In dem Debug-Fenster können Sie Folgendes:

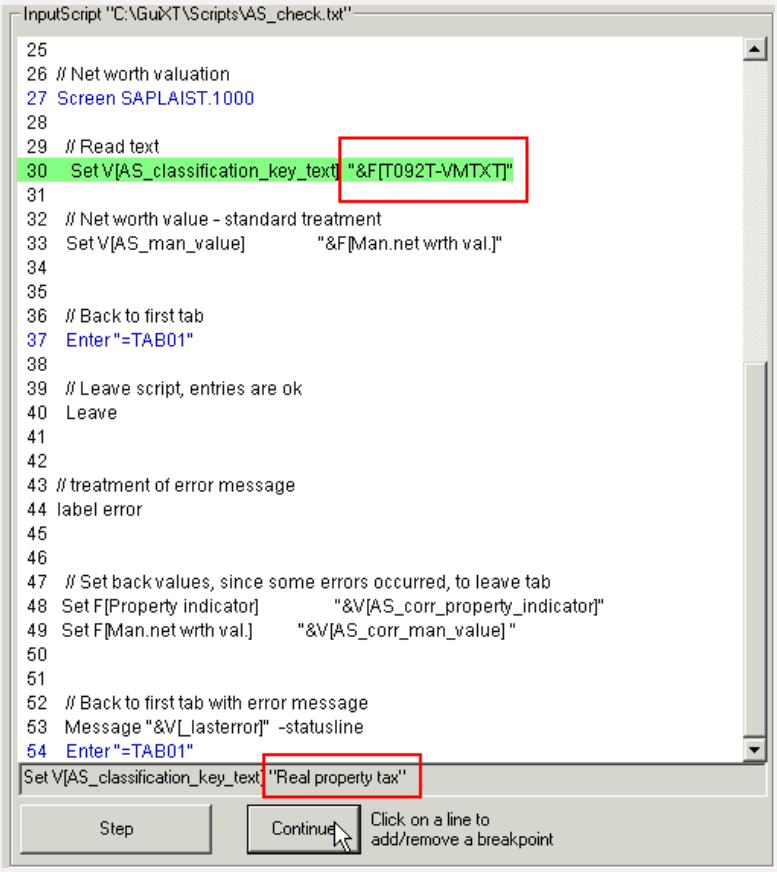
- Inhalt jeder einzelnen Variable Ansehen
- Expressionen Auswerten die Variables, Parameter oder Feldwerte enthalten
- InputScript Schritt für Schritt Ausführen
- Breakpoints Setzen



Die active Scriptzeile (also die Zeile die als Nächste ausgeführt wird) wird gelb dargestellt. Um einen Breakpoint zu setzen brauchen Sie nur auf die Zeile zu klicken. Die Zeile wird dann auf türkisfarbigem Hintergrund dargestellt. Klicken Sie erneut auf die Zeile um den Breakpoint wieder zu löschen.

Breakpoints können auf jeden ausführbaren Ausdruck sowie auf **screen**-Anweisungen gesetzt werden aber nicht auf leere Zeilen oder Kommentar-Zeilen.

Durch Klicken auf eine Variable wird diese zur "Expressions"-Liste hinzugefügt. Sie können hier auch andere Ausdrücke hinzufügen, z.B. durch Verwendung der Substring-Notation &V[abc](50-100), oder Parameter wie &U[ABCD] anzeigen. Sie können auch verschachtelte Ausdrücke wie "&V[QTY_&V[i]]" benutzen.



Wird ein Breakpoint erreicht, wird die Zeile grün dargestellt. Für jede aktive Zeile wird der Debugger die Zeile nach Auswertung aller &-Ausdrücke zeigen. Zum Beispiel anstelle von "&F[T092T-VMTXT]", wie im InputScript geschrieben, wird der Text "Real property tax" angezeigt (Zeile direkt über den Step/Continue-Tasten).

Jeder Syntaxfehler wir als Popup-Nachricht während der Schritt-für-Schritt-Ausführung angezeigt.

3.5. GuiXT Script Editor

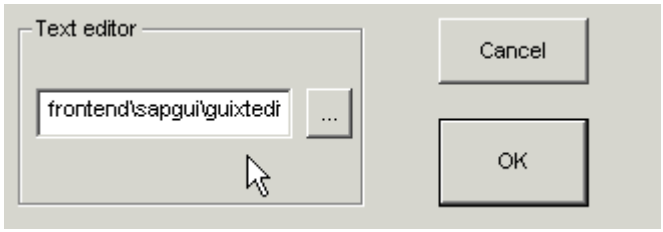
3.5.1. Übersicht

Der GuiXT Script Editor unterstützt

- Scripte sowohl im ANSI- als auch im UTF-8 Zeichensatz
- syntaxgesteuerte Farben
- integrierte F1-Hilfe für GuiXT Schlüsselwörter
- einfache Navigation zu "Include" und "process=" Dateien
- Pretty Print (automatisches Einrücken)
- Auskommentieren von Code-Blöcken
- Einfügen von Befehlsmustern
- Eigene Befehlsmuster
- Schreiben von Backup-Dateien

Insbesondere komplexere InputScripts sind so leichter zu bearbeiten.

3.5.2. Installation

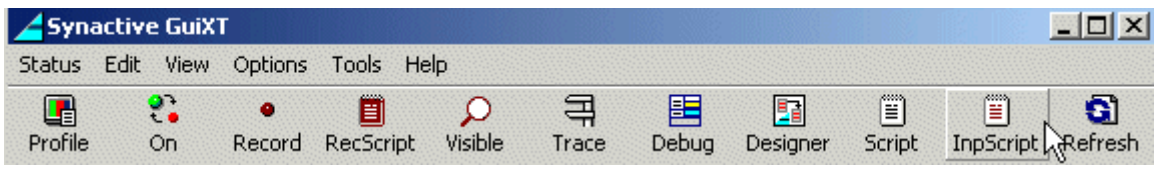


Bitte laden Sie das Editor Setup Programm (guixt.editor.setup.exe) von unserer Download Seite herunter und führen es aus. Es kopiert das Programm guixteditor.exe in das von Ihnen gewählte Installationsverzeichnis. Zusätzlich trägt es den Script Editor als Standardeditor für Scripts in das GuiXT Profile ein. Sie können diese Angabe im GuiXT Profile jederzeit wieder überschreiben und "notepad.exe" oder einen anderen Editor Ihrer Wahl eintragen.

Es ist empfehlenswert, auch die html Hilfedatei (guixt_d.chm für Deutsch) herunterzuladen und in das gleiche Verzeichnis zu kopieren. Die integrierte F1-Hilfe des Editors arbeitet mit dieser Datei.

Falls Sie später eine neue Version des Editors oder der Hilfedatei herunterladen, kann das auch unabhängig voneinander geschehen.

Nach Installation des Editors wird bei Klick auf "Script" oder "InputScript" in GuiXT der Script Editor aktiv.

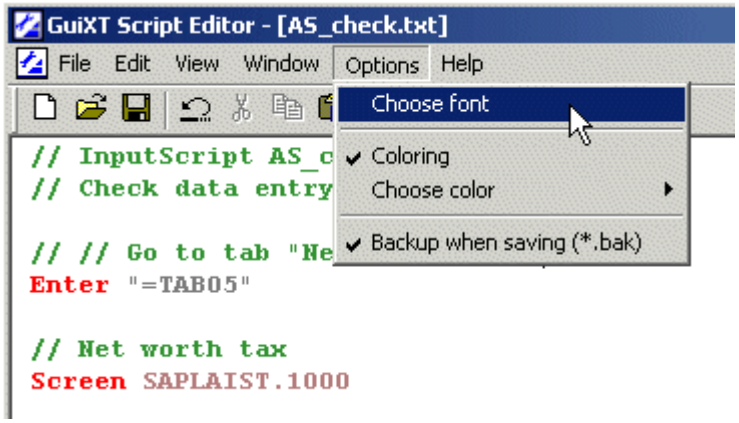

Synactive GuiXT
Status Edit View Options Tools Help
Profile On Record RecScript Visible Trace Debug Designer Script InpScript Refresh

3.5.3. ANSI und UTF-8 Format

Für SAP Unicode-Systeme müssen Sie die Scripte im UTF-8 Format anlegen, andernfalls werden manche Zeichen, zum Beispiel Umlaute, nicht korrekt dargestellt und Screenelemente mit Sonderzeichen nicht erkannt. Der Scripteditor unterstützt Scripte in beiden Formaten. Wenn Sie den Zeichensatz eines Scripts ändern möchten, können Sie im "Speichern unter" Dialog das gewünschte Format explizit wählen. Neue Scripte werden jeweils in dem zuletzt verwendeten Zeichensatz angelegt. In der Titelleiste wird der Zeichensatz zusätzlich zu dem Dateinamen angezeigt.

3.5.4. Editor-Einstellung Zeichsatz

Über **Options->Choose font** können Sie den gewünschten Zeichensatz und die Schriftgrösse einstellen. Für unsere Beispielen hier haben wir



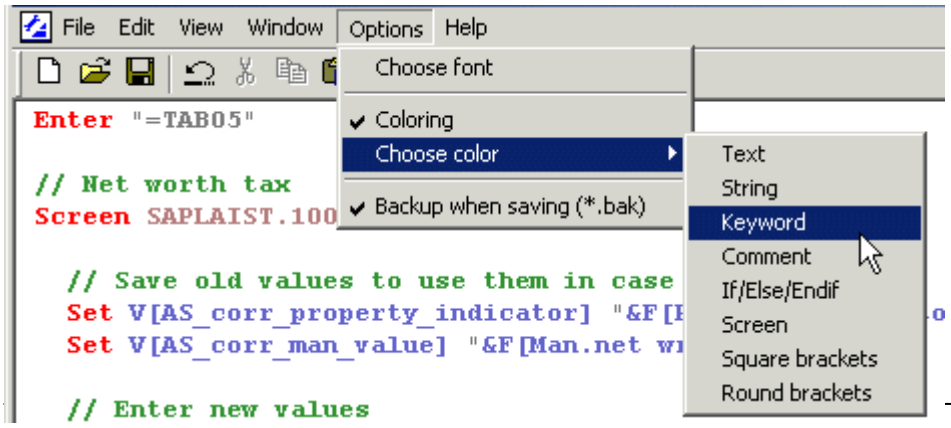
Courier New, Fett, Schriftgrad 10

gewählt. Es ist empfehlenswert, mit einer Schrift fester Zeichengrösse (fixed-font) zu arbeiten, da sonst einzelne Teile Ihrer Skripte vielleicht nicht mehr wie gewünscht ausgerichtet angezeigt werden. Beispiel:

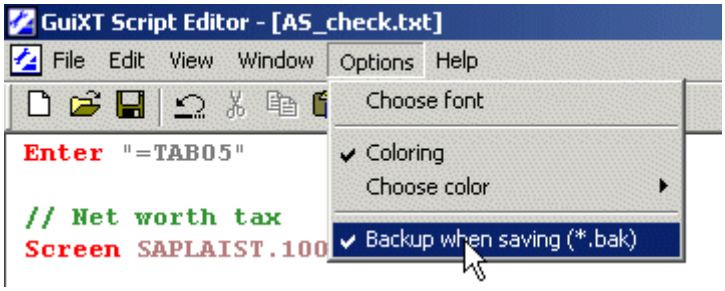
```
// Enter new values
Set F[Classification key]      "&V[AS_classification_key]"
Set F[Property indicator]      "&V[AS_property_indicator]"
Set F[Reason for man.val.]     "&V[AS_man_reason]"
Set F[Man.net wrth val.]       "&V[AS_man_value]"
```

3.5.5. Editor-Einstellung Farben

(falls Sie mit den Standard-Einstellungen nicht zufrieden sind...)



3.5.6. Editor-Einstellung Backup Dateien



Wenn Sie eine Datei im Editor sichern, z.B. die Datei "C:\GuiXT\Scripts\va02_zusatzdaten.txt", wird bei eingeschalteter Backup-Funktion automatisch zunächst eine Kopie der existierenden Datei erstellt. Der Name wird aus dem Originalnamen durch Anhängen von Datum, Zeit und ".bak" gebildet, in unserem Beispiel also

"C:\GuiXT\Scripts\va02_zusatzdaten.txt.20050221.163105.bak"

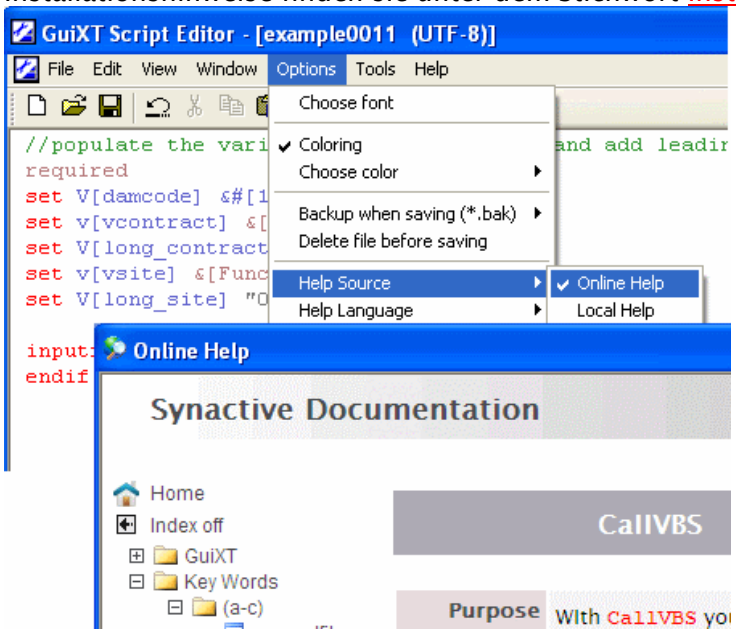
Für die Zukunft ist im Editor ein Vergleichstool für Backup-Dateien geplant.

3.5.7. Hilfe Online oder Lokal

Sie haben bei der Auswahl für die Quelle der Hilfe zwei Optionen:

Es wird entweder eine Onlineversion in einem neuen Fenster angezeigt, welche direkt von der Synactive-Website geladen wird. (Hierfür ist eine bestehende Internetverbindung erforderlich.)

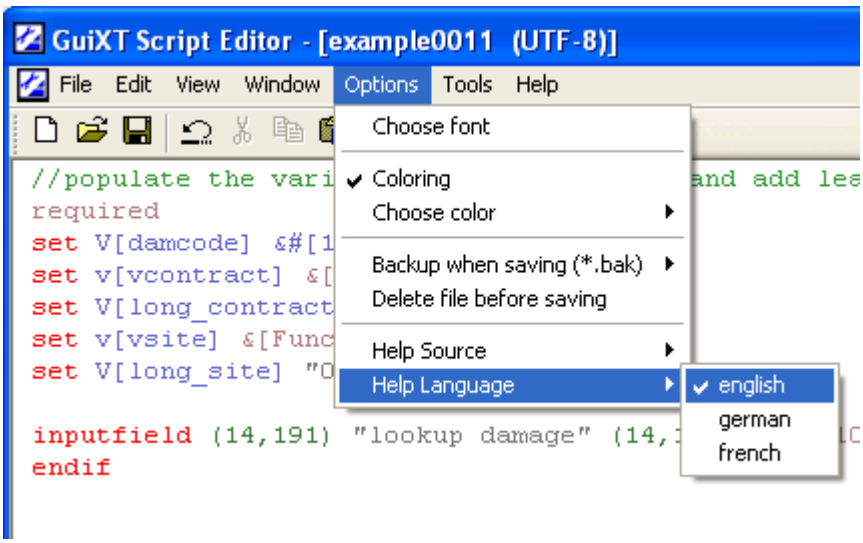
Oder es wird auf eine lokal installierte Datei (guixt_d.chm für deutsch) zugegriffen.
Installationshinweise finden sie unter dem Stichwort Installation.



3.5.8. Hilfe Sprachauswahl

Die Hilfe steht Ihnen in drei Sprachversionen zur Verfügung:
In englischer, deutscher oder französischer Sprache.

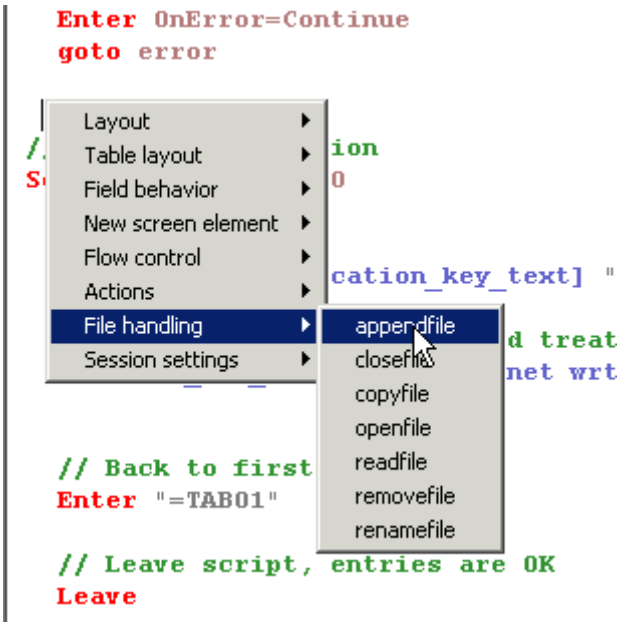
Diese können Sie unter dem Menüpunkt **Help Language** festlegen.



3.5.9. Befehlsmuster einfügen

Um ein Befehlsmuster einzufügen, klicken Sie entweder auf den "insert command" Button in der Drucktastenleiste:

Oder Sie klicken mit der echten Maustaste in eine leere Editor-Zeile. Es erscheint dann ein zweistufiges Menü, das sämtliche GuiXT Schlüsselwörter enthält:



Wenn Sie ein Schlüsselwort auswählen, wird im Editor in einer oder in mehreren Zeilen eine typische Form der ausgewählten Anweisung vorgeschlagen. Eventuell sind Bemerkungen zur Anweisung oder Zusatzoptionen als Kommentar enthalten. Beispiel:

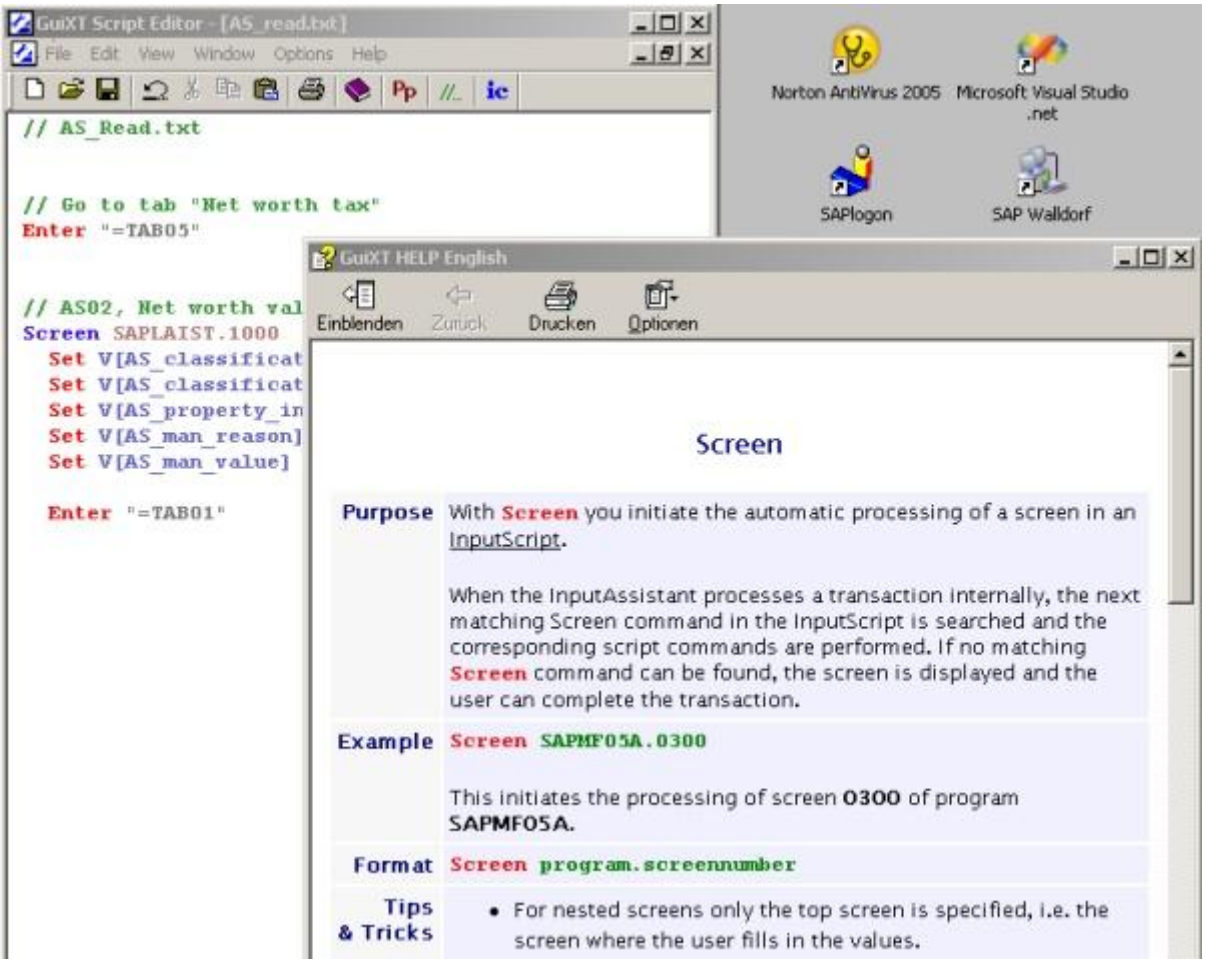
```
BoxSize G[Box] (rows,columns) // if rows=0 or columns=0: existing value is retained
```

3.5.10. Eigene Befehlsmuster einfügen

Sie können sich eigene Befehlsmuster definieren, die Sie genauso einfügen können, zum Beispiel einen standardisierten Kopf für die Scripte oder einen Korrekturvermerk. Legen Sie dazu ein Unterverzeichnis "mytemplates" im Installationsverzeichnis des Scripteditors an, also in dem Verzeichnis, in dem sich auch die Datei "guixteditor.exe" befindet. Um ein eigenes Befehlsmuster zu definieren, müssen Sie nun nur eine ".txt" Datei mit dem gewünschten Inhalt in dem "mytemplates" Verzeichnis hinzufügen. Der Dateiname ist beliebig, sollte aber aussagekräftig sein, da er Ihnen bei "ic" unter "my templates" vom Editor angezeigt wird.

3.5.11. Schlüsselwort Hilfe anzeigen

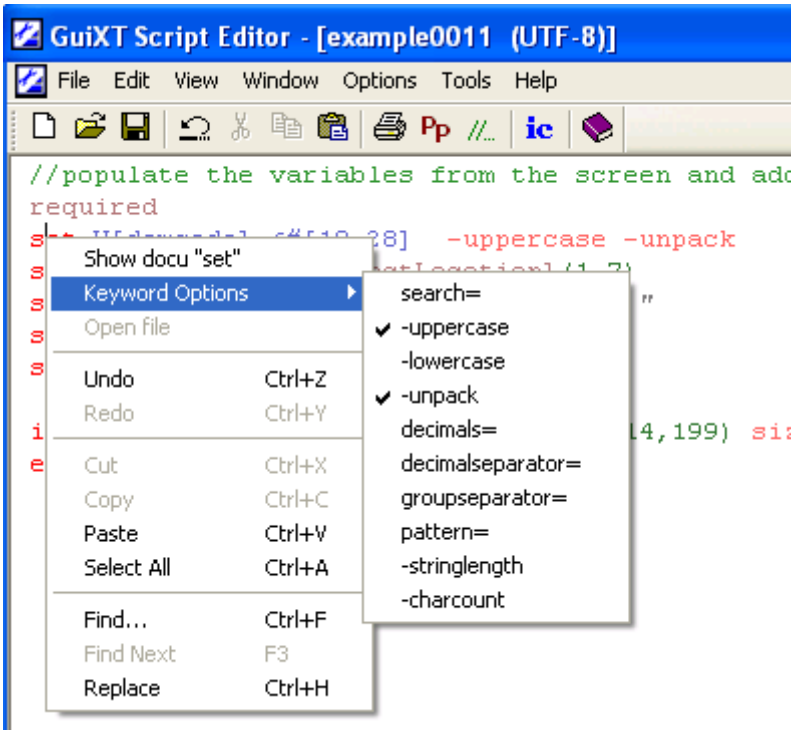
Das geschieht über F1 in der gewünschten Zeile, oder über das rechte Maustasten-Menü:


GuiXT Script Editor - [AS_read.txt]
File Edit View Window Options Help
// AS_Read.txt
// Go to tab "Net worth tax"
Enter "=TAB05"
// AS02, Net worth val
Screen SAPLAIST.1000
Set V[AS_classificat
Set V[AS_classificat
Set V[AS_property_in
Set V[AS_man_reason]
Set V[AS_man_value]
Enter "=TAB01"
Norton AntiVirus 2005
Microsoft Visual Studio .net
SAPlogon
SAP Walldorf
GuiXT HELP English
Einblenden
Zurück
Drucken
Optionen
Screen
Purpose
With Screen you initiate the automatic processing of a screen in an InputScript.
When the InputAssistant processes a transaction internally, the next matching Screen command in the InputScript is searched and the corresponding script commands are performed. If no matching Screen command can be found, the screen is displayed and the user can complete the transaction.
Example
Screen SAPMF05A.0300
This initiates the processing of screen 0300 of program SAPMF05A.
Format
Screen program.screennumber
Tips & Tricks
For nested screens only the top screen is specified, i.e. the screen where the user fills in the values.

3.5.12. Optionen zu einem Schlüsselwort anzeigen

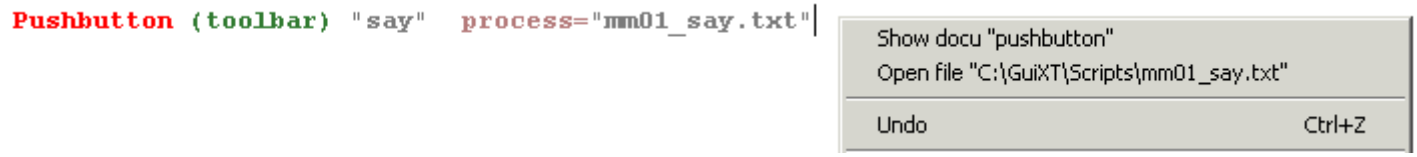
Das geschieht über F4 in der gewünschten Zeile, oder über das rechte Maustasten-Menü. Durch Auswahl wird die entsprechende Option in das Skript eingefügt, falls sie nicht bereits vorhanden ist. Dies ist durch

einen Haken im Optionenmenü gekennzeichnet.

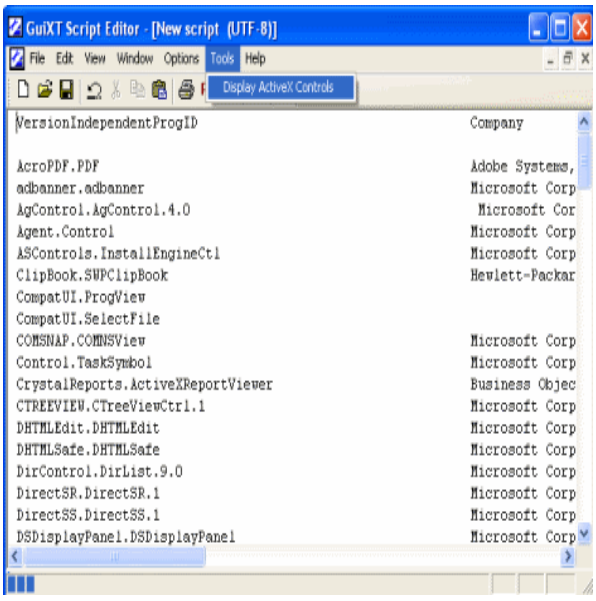


3.5.13. Include oder Process Datei öffnen

Entweder durch Drücken von F2 in der Zeile, oder über das rechte Maustasten-Menü:

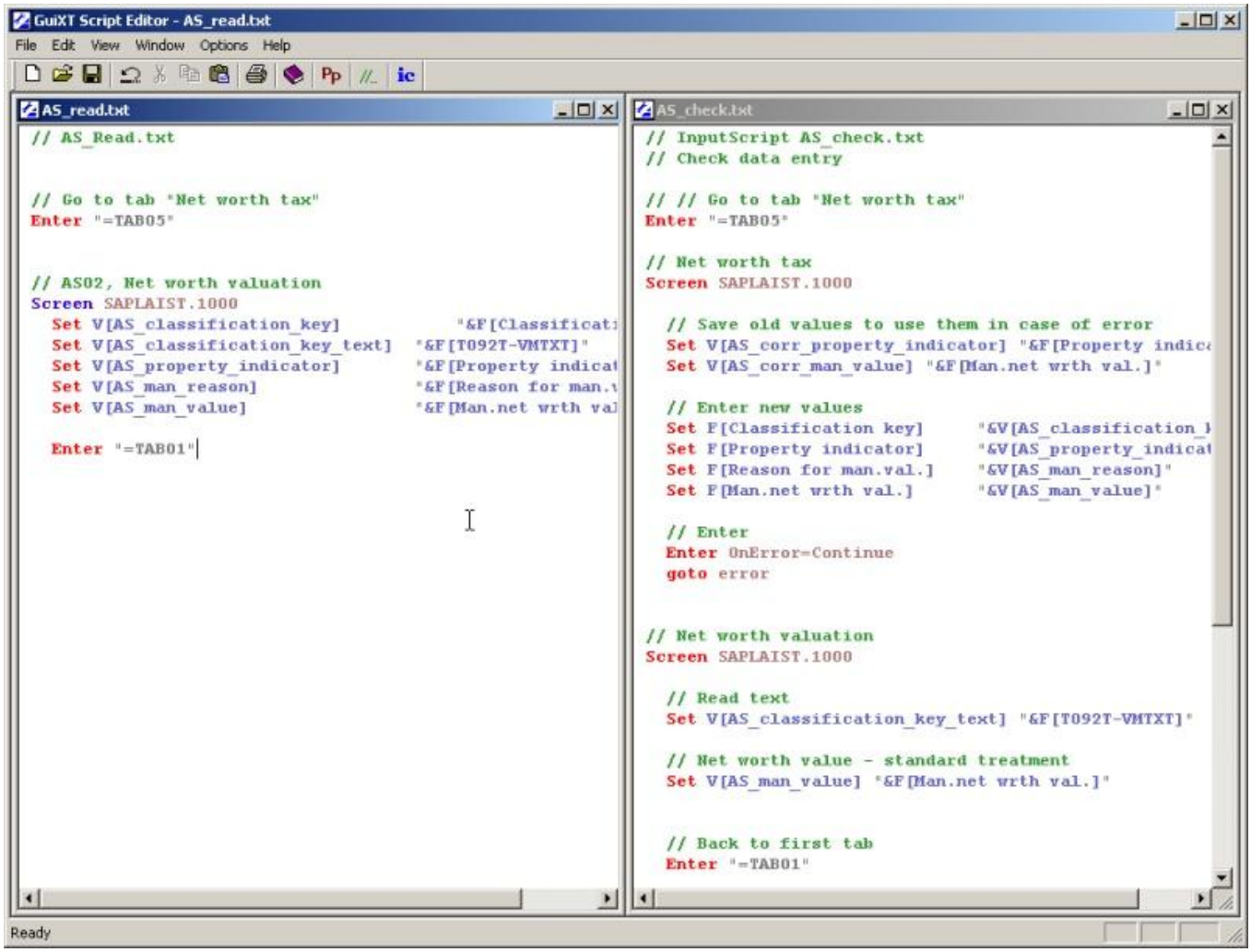


3.5.14. ActiveX Controls anzeigen



3.5.15. Mit mehreren Dateien arbeiten

Die Option "tile vertically" im "Window" Menü ist hierbei oft praktisch:



3.5.16. Weitere Funktionen

Weitere Funktionen, wie "pretty print" oder das Auskommentieren von Code-Blöcken probieren Sie am besten selbst aus. Daneben stehen die üblichen Editor-Funktionen (Undo, Cut&Paste, Drucken) zur Verfügung.

4. Befehle im Detail (GuiXT)

4.1. Box

Funktion	Mit der **Box**-Anweisung können Sie einen rechteckigen Rahmen um eine Gruppe anderer Bildelemente zeichnen.
Beispiele	`Box (10,20) (16,44) "Rechnungsdaten"` Es wird ein rechteckiger Rahmen gezeichnet mit dem linken oberen Eckpunkt `(10,50)` und dem rechten unteren Eckpunkt `(16,44)`. In den Rahmen wird der Titel `"Rechnungsdaten"` eingeblendet.

Format	`Box (Zeile1,Spalte1) (Zeile2,Spalte2)`
	
	`Box (Zeile1,Spalte1) (Zeile2,Spalte2) "Rahmentitel"`

Tipps & Tricks	• In den Rahmentitel können Bildvariablen eingestreut werden, z.B.
	
	`Box (10,20) (16,44) "Rechnungsdaten zu &[Referenznummer]"`
	
	• Falls in einen Rahmen über **Image** eine Abbildung eingefügt wird, sollten Sie aus optischen Gründen in der **Image** Anweisung etwas größere Koordinaten für die linke obere Ecke wählen, z.B.:
	
	`Box (10,20) (16,44) "Rahmentitel"`
	`Image (10.2,20.5) (16,44) "Bilddatei"`
	
	• Beispiele finden Sie auch in den GuiXT Tutorials 2 und 3 und in den Designer Tutorials 6 und 7.

4.2. Boxsize

Funktion	Mit der **BoxSize**-Anweisung können Sie die Größe eines Gruppenrahmens ändern.
Beispiele	`BoxSize G[Allgemeine Daten] (5,40)`
Format	`BoxSize P[Box] (Zeilen,Spalten)` **Zeilen** ist dabei die gewünschte Höhe und **Spalten** die Breite des Rahmens. Falls einer der beiden Werte 0 ist, wird die bisherige Anzahl Zeilen bzw. Anzahl Spalten beibehalten.
Tipps & Tricks	Beispiele finden Sie im GuiXT Tutorial 1.

4.3. ButtonSize

Funktion	Mit der `ButtonSize`-Anweisung können Sie die Größe einer Drucktaste (Pushbutton) ändern.
Beispiele	`ButtonSize P[Anzeige] (3,20)`
Format	`ButtonSize P[Pushbutton] (Zeilen,Spalten)` `Zeilen` ist dabei die gewünschte Höhe und `Spalten` die Breite der Drucktaste
Tips &Tricks	`ButtonSize` wird vorerst nicht im SapGui for HTML unterstützt.

4.4. ColumnHeader

Funktion	Mit der `ColumnHeader`-Anweisung können Sie die Spaltenüberschriften einer Tabelle (Table Control) ändern.
Beispiele	`ColumnHeader [Material] "Artikel"` `ColumnHeader [Table.2,Material] "Artikel"`

Format	`ColumnHeader [Spalte] "text"`
	`ColumnHeader [Tabelle,Spalte] "text"`
	Die explizite Angabe der Tabelle ist nur nötig, wenn zwei oder mehr Tabellen angezeigt werden und der Spaltenname nicht eindeutig ist. Die Tabelle wird dann über ihren Titel benannt, falls ein Titel vorhanden ist, sonst als Table, Table.2, Table.3,... Am besten schauen Sie in GuiXT unter **View->Screen elements** nach, wie die Tabelle heißt.

Ikonen und Quickinfo	Die Spaltenüberschrift kann durch eine SAP Ikone und eine Quickinfo ergänzt werden. Das geschieht in der SAP-Konvention
	`ColumnHeader [Spalte] "@id\Qquickinfo@text"`
	Hier ist `id` das Kennzeichen der SAP-Ikone, `quickinfo` der als Quickinfo (Tooltip) anzuzeigende String, `text` die Bezeichnung der Spalte. Sie finden eine Liste aller SAP Ikonen in der Transaktion ICON, leider dort ohne die internen Kennzeichen der Ikonen. Diese sind in dem ABAP Programm <ICON> abgelegt (Transaktion SE38, Programmname <ICON> eingeben, Quelltext anzeigen)
	Beispiel:
	`ColumnHeader [Beschreibung] "@80\QBitte geben Sie eine Beschreibung ein@Beschreibung"`

Tips & Tricks	• Sie können die Ikone und die Quickinfo angeben, um z.B. die Spalte zu markieren. • Lesen Sie bitte auch das Special Topic Anpassen von Tabellen.

4.5. ColumnOrder

Funktion	Mit der `ColumnOrder`-Anweisung können Sie die Anordnung der Tabellenspalten (Table Control) ändern.
Beispiele	`ColumnOrder [Material] 2` `ColumnOrder [Table.2,Material] 2`

Format	`ColumnOrder [Spalte] n`
	`ColumnOrder [Tabelle,Spalte] n`
	
	`n` ist dabei die gewünschte neue Nummer der Spalte, d.h. bei n=1 wird die Spalte als erste Spalte angezeigt, bei n=2 als zweite.
	
	Die explizite Angabe der Tabelle ist nur nötig, wenn zwei oder mehr Tabellen angezeigt werden und der Spaltenname nicht eindeutig ist. Die Tabelle wird dann über ihren Titel benannt, falls ein Titel vorhanden ist, sonst als Table, Table.2, Table.3,..., Am besten schauen Sie in GuiXT unter **View->Screen elements** nach, wie die Tabelle heißt.
	
	Die Position der übrigen Spalten bleibt unverändert.
	Die Angabe im GuiXT-Skript überlagert eventuelle andere Einstellungen der Tabelle.

4.6. Columnsize

Funktion	Mit der `ColumnSize`-Anweisung können Sie die Breite einer Tabellenspalte (Table Control) ändern.
Beispiele	`ColumnSize [Material] 8` `ColumnSize [Table.2,Material] 8`

Format	`ColumnSize [Spalte] n`
	`ColumnSize [Tabelle,Spalte] n`
	
	`n` ist dabei die gewünschte Spaltenbreite (Anzahl Zeichen) bei Ein- und Ausgabe.
	Die explizite Angabe der Tabelle ist nur nötig, wenn zwei oder mehr Tabellen angezeigt werden und der Spaltenname nicht eindeutig ist. Die Tabelle wird dann über ihren Titel benannt, falls ein Titel vorhanden ist, sonst als Table, Table.2, Table.3,..., Am besten schauen Sie in GuiXT unter **View->Screen elements** nach, wie die Tabelle heißt.
	
	`ColumnSize [...] 0`: Bei Angabe der Spaltenbreite 0 wird die Spalte gänzlich unterdrückt.
	
	Die Angabe der Spaltenbreite im GuiXT-Skript überlagert eventuelle andere persönliche Einstellungen der Tabelle.

Tips &Tricks	• Im Gegensatz zu `ColumnWidth` wird nicht nur die visuelle Spaltenbreite geändert: eventuelle längere Werte werden abgeschnitten und die Eingabe ist nur in der neuen Länge möglich.

	• Bei Einsatz von SAP GUI 4.6D können einige sehr große SAP Masken (große Tabellen) nicht um zusätzliche GuiXT-Elemente erweitert werden. Es erfolgt dann eine entsprechende Fehlermeldung im Script-Fenster von GuiXT. Sie können dann durch `ColumnSize` Angaben den für das Bild benötigten Platz verkleinern, indem Sie nicht benötigte Spalten weglassen oder andere in der Länge reduzieren. Ab SAP GUI 6.20 tritt dieses Problem nicht mehr auf.

4.7. ColumnWidth

Funktion	Mit der `ColumnWidth`-Anweisung können Sie die Breite einer Tabellenspalte (Table Control) ändern.
Beispiele	`ColumnWidth [Material] 8` `ColumnWidth [Table.2,Material] 8`

Format	`ColumnWidth [Spalte] n`
	`ColumnWidth [Tabelle,Spalte] n`
	
	`n` ist dabei die gewünschte sichtbare Spaltenbreite (Anzahl Zeichen). Längere Felder dieser Spalte werden automatisch scrollbar.
	Die explizite Angabe der Tabelle ist nur nötig, wenn zwei oder mehr Tabellen angezeigt werden und der Spaltenname nicht eindeutig ist. Die Tabelle wird dann über ihren Titel benannt, falls ein Titel vorhanden ist, sonst als Table, Table.2, Table.3,... Am besten schauen Sie in GuiXT unter **View->Screen elements** nach, wie die Tabelle heißt.
	
	`ColumnWidth [...] 0`: Bei Angabe der Spaltenbreite 0 wird die Spalte gänzlich unterdrückt, kann aber mit der Maus wieder angezeigt werden (im Gegensatz zum Befehl `ColumnSize [...] 0`).
	
	Die Angabe der Spaltenbreite im GuiXT-Skript überlagert eventuelle andere persönliche Einstellungen der Tabelle.

4.8. Comment

Funktion	Mit der `Comment`-Anweisung können Sie einen beliebigen Kommentartext auf dem SAP-Bild anzeigen
Beispiele	`Comment (10,50) "Bitte benutzen Sie zum Drucken"` `Comment (11,50) "das Formular M10A"`
Format	`Comment (Position) "Text"`
Tips &Tricks	• Zum Einfügen von optisch ansprechend gestalteten Hinweisen können Sie die `Image`-Anweisung nutzen , oder Sie können mit der `View`-Anweisung rtf-Texte (aus Word z.B. erzeugbar) oder html-Dateien im SAP-Bild anzeigen

4.9. Compress

Funktion	Mit der `Compress`-Anweisung können Sie Leerzeilen in einem Gruppenrahmen oder in dem gesamten Bild löschen. Die darunterliegenden Zeilen werden nach oben geschoben und die Rahmenlänge wird entsprechend verringert. Das ist insbesondere dann sinnvoll, wenn Sie in einem Bild viele Felder gelöscht haben und ein kompaktes Layout erzielen wollen, ohne allzuviel umzugruppieren. Es findet nur eine vertikale Komprimierung statt (Zeilen), keine horizontale (Spalten).
Beispiel	`Compress G[Weitere Daten]`

Format	`Compress G[Gruppenrahmen]`
	Leerzeilen innerhalb des angegebenen Gruppenrahmens werden gelöscht, der Rahmen entsprechend verkürzt.
	`Compress`
	Leerzeilen im gesamten Bild werden gelöscht, Gruppenrahmen verkürzt.

Tips &Tricks	• Löschen Sie die nicht benötigten Felder in Ihrem Script, bevor Sie das Bild komprimieren. • Beim Löschen ist unter Umständen die `"-triple"` Option nötig, um Textfelder hinter einem Eingabefeld ebenfalls zu löschen. Andenfalls wird die Zeile später nicht komprimiert. • Sie können Gruppenrahmen oder Felder vor oder nach dem Komprimieren verschieben. Meist ist es aber sinnvoll, das vor dem Komprimieren zu tun, da sonst die Effekte schwer überschaubar sind (Überlagerungen). • Bei dem Komprimieren eines einzelnen Gruppenrahmens bezieht sich "Leerzeile" nur auf den Bereich innerhalb des Rahmens, ansonsten auf den gesamten Bereich. Es kann deshalb sinnvoll sein, zunächst einzelne Gruppenrahmen zu komprimieren, dann erst das gesamte Bild. • Eigene Elemente (Texte, Drucktasten, Eingabefelder,...) bitte vor dem Komprimieren anlegen, da sonst dafür kein Platz freigehalten wird. • Eigene Elemente werden ebenfalls unter Umständen bei dem Komprimieren verschoben und eigene Gruppenrahmen verkleinert.

4.10. Default

Funktion	Mit der `Default`-Anweisung können Sie Vorschlagswerte in Eingabefelder setzen. Falls das Feld bei der Anzeige des Bildes leer ist, wird der von Ihnen gesetzte Wert dem Benutzer vorgeschlagen. Ein von SAP gesetzter Wert wird nicht überschrieben.
Beispiel	`Default F[Auftragsart] "BV"` Das Feld Auftragsart erhält, falls es noch initial ist, den Wert `"BV"`.
Format	`Default F[Eingabefeld] "Wert"`

Tips &Tricks	• Auch bei Feldern, die Sie durch `del` ausblenden, können Sie einen Vorschlagswert (der in diesem Fall ein konstanter Wert ist) setzen. • Falls Sie ein Feld über die `Radiobutton` Anweisung als eine Reihe von Auswahlknöpfen darstellen, wirkt sich das Setzen eines Vorschlagswertes durch `Default` so aus, daß der betreffende Knopf aktiviert ist.

4.11. Del

Funktion	Mit der `del`-Anweisung können Sie Felder, Feldgruppen, Drucktasten, Auswahlknöpfe, Tabellen, Ankreuzfelder und Menüeinträge aus dem Bild entfernen.

Beispiele	`del F[PartnerGsber.]`
	`del G[Mandantenspezifische Konfiguration]`
	`del #[12,60]`
	`del F[PartnerGsber.]+(1,3)`
	`del M[Ändern]`
	`delM[UPD]`
	`delM[1,4,2]`

Format	`del [Bildelement]`
	Bei Feldern wird sowohl die Feldbezeichnung als auch der dahinter stehende Wert bzw. das dahinter stehende Eingabefeld gelöscht. Bei Feldgruppen werden alle Elemente innerhalb des Gruppenrahmens und der Gruppenrahmen selbst gelöscht.
	`del M[Menüeintrag]`
	`delM[fcode]`
	`delM[i,j,k,l]`
	Löschen von Menüeinträgen. Sie können entweder den Text aus dem Menü oder den internen Funktionscode (angezeigt bei F1) verwenden. In Sonderfällen kann auch die hierarchische Position des Menüeintrags, jeweils beginnend bei 1, angegeben werden. GuiXT löscht dann
	• die Zeile im Menü
	• die entsprechende Drucktaste in der Drucktastenleiste, falls vorhanden
	• die Zuordnung zu einer F-Taste oder Tastenkombination
	• Falls der Eintrag ein Untermenü ist, werden die Einträge des Untermenüs in gleicher Weise gelöscht
	Ebenfalls verhindert wird die direkte Eingabe des entsprechenden Funktionscodes durch den Benutzer im OK-Code-Feld.
	Dokumentation dazu siehe auch in den "Special Topics": Löschen von Menüeinträgen

Zusatzoptionen

`-Triple`	Sinnvoll bei Ein-/Ausgabefeldern mit einem dahinter angezeigten Klartext. Auch dieses dritte Feld wird gelöscht.
`-Value`	Möglich bei Ein-/Ausgabefeldern mit vorangestelltem Text. Es wird das Feld, nicht jedoch der Text gelöscht.
`-Text`	Nur sinnvoll bei Ein-/Ausgabefeldern mit vorangestelltem Text. Es wird lediglich der Text, nicht das Feld selbst gelöscht.
`-Box`	Nur sinnvoll, wenn das Bildelement eine Feldgruppe ist. Es wird lediglich der Rahmen um die Feldgruppe herum gelöscht, die Felder selbst bleiben erhalten.
`-WithBox`	Nur bei del S[tabstrip] sinnvoll. Es wird der Rahmen um die Felder ebenfalls gelöscht.

Tipps & Tricks

- Falls man alle Felder innerhalb eines Gruppenrahmens bis auf eins löschen möchte, kann man das auszusparende Feld zunächst durch `pos` aus dem Gruppenrahmen heraus an eine andere Position verschieben und anschließend die Feldgruppe löschen.
- Auch Drucktasten aus der Drucktastenleiste können durch `del` gelöscht werden. Falls die Drucktaste nur aus einer Ikone (ohne Text) besteht, benennt man sie über den Quickinfo-Text.
- Sie können auch mehrere Bildelemente löschen:
  `del [Bildelement1] [Bildelement2]`
  Es wird das Bildelement oben links und das Bildelement unten rechts angegeben. Alle Bildelemente zwischen diesen werden gelöscht.
- Um aus der Toolbar solche Drucktasten bzw. Pushbuttons zu löschen, die keinen Text, sondern nur eine Ikone enthalten, haben Sie folgende Möglichkeiten:

```
// Del mit Angabe der Funktionstaste (F-key)
delP[/8]

// Del mit Angabe des Quickinfo-Textes
delP[Single test]

// Del mit Angabe der Icon id
delP[@12@]

// Löschen des ensprechenden Menübefehls zusammen mit dem
Pushbutton
delM[TEST]
```

4.12. Disable

Funktion	Mit der `Disable`-Anweisung können Sie spezielle GuiXT-Funktionen aussschalten.
Beispiel	`Disable auto return`
Format	`Disable auto return` Unmittelbar nach Ablauf eines InputScripts interpretiert GuiXT das Drücken von F15 (gelbe Exit-Ikone) als Rückkehr zu der ursprünglichen Transaktion. Durch `Disable auto return` im InputScript wird dieser Automatismus ausser Kraft gesetzt.

4.13. FieldSize

Funktion	Mit der `FieldSize`-Anweisung können Sie die Länge eines Eingabe- oder Ausgabefeldes verändern.
Beispiel	`FieldSize F[Material] 10`
Format	`FieldSize [Bildelement] n` Das angegeben Feld kann ein Eingabefeld oder Ausgabefeld sein. Die Feldlänge wird auf `n` Zeichen gesetzt; dabei kann `n` ein Wert zwischen 1 und der ursprünglichen Feldlänge sein.
Zusatzoptionen	`-Scrollable` Nur die visuelle Eingabelänge des Feldes wird reduziert, das Feld wird scrollbar und kann in der ursprünglichen Länge eingegeben werden.
Tipps & Tricks	• Falls der anzuzeigende Wert mehr als `n` Zeichen hat, wird das Feld automatisch scrollbar, sodaß der Benutzer auch diesen längeren Wert anzeigen kann. Im allgemeinen sollte `FieldSize` aber nur dann genutzt werden, wenn die Feldinhalte immer aus maximal `n` Zeichen bestehen, oder mit der Option `-Scrollable` kombiniert werden.

4.14. File

Funktion	Mit der `File`-Anweisung können Sie SAPWR Skripte (SAP Web Repository) mit langen Namen einen kurzen Alias-Namen zuweisen. Dies kann sinnvoll sein, da es in SAPWR eine Beschränkung auf Namen mit höchstens 30 Zeichen gibt.
Beispiele	`File "saplsmtr_navigation.e0100.txt" Alias="menu.txt"`
Format	`File "langer_Dateiname" Alias="kurzer_Name"`
Tips &Tricks	Die `File`-Anweisung muss im Session Skript stehen, also DSession.txt, ESession.txt etc.

4.15. GlobalTextreplace

Funktion	Mit der `GlobalTextReplace`-Anweisung können Sie durch eine einzige Anweisung Texte bei der Anzeige sämtlicher SAP Masken ersetzen.
Beispiel	`GlobalTextReplace "Buchungskreis" "Landesgesellschaft"`
Format	`GlobalTextReplace "alt" "neu"` In folgenden Bildelementen wird der String `"alt"` durch `"neu"` ersetzt: • Feldbezeichnungen • Bildtitel • Rahmentitel

	• Drucktastentexte und Quickinfo • Nachrichtenzeile • Menütexte • Spaltenüberschriften in Tabellen • Tabellentitel Die Ersetzung wird nach folgenden Regeln durchgeführt: 1. Groß/Kleinschreibung wird ignoriert, jedoch wird der erste Buchstabe im neuen Text groß geschrieben, wenn er im alten Text ebenfalls groß geschrieben war 2. Längere Texte werden vor kürzeren Texten ersetzt **Die Anweisung `GlobalTextReplace` sollte in das Skript DSession.txt gelegt werden. Dieses Skript wird nach dem logon pro SAP Session (und pro Alternativmodus) genau einmal durchgeführt.**

Tips &Tricks	• Die Anweisung `TextReplace` leistet die gleiche Textersetzung für ein einzelnes Bild
	• Im Deutschen sind die unterschiedlichen Wortendungen ("Buchungskreis**e**", aber "Landesgesellschaft**en**") manchmal ein Problem. Sie können sich helfen durch weitere `GlobalTextReplace`-Anweisungen, z.B.
	`GlobalTextReplace "Buchungskreise" "Landesgesellschaften"`
	oder durch `TextReplace` Angaben für den ganzen Text, z.B.
	`TextReplace "Auswahl des Buchungskreises" "Auswahl der Landesgesellschaft"`
	• Siehe auch das Special Topic "Unternehmensspezifische Begriffe verwenden – Generelle Umbenennungen von SAP Anzeigetexten "

4.16. Icon

Funktion	Mit der `Icon`-Anweisung können Sie die Ikone in der Titelzeile des SAP Systems dauerhaft ändern. Sie können somit z.B. zwei SAP Anwendungen auf einen Blick unterscheiden, auch wenn die Fenster nur in der Taskleiste erscheinen.
Beispiel	`Icon "C:\Icones\guixt.ico"`
Format	`Icon "Name der Datei.ico"` Die Ikone erscheint anstelle der SAP Ikone in der oberen linken Ecke der SAP Bildschirme oder in der Taskleiste vor dem Programmnamen. Sie können die Ikone im Format 16x16 oder 32x32 erstellen.
Tips &Tricks	• Geben Sie die `Icon`-Anweisung direkt in das Skript des Logon-Bildschirms ein. Somit erscheint die Ikone in allen weiteren SAP Bildschirmen

- Falls Sie die Ikone abhängig vom Mandanten auswählen wollen, können Sie die `Icon`-Anweisung in dem Skript des SAP Systemmenüs S000 definieren
- Sie können die Ikone auch als "transparent" definieren, die Hintergrundfarbe ist dann die der Titelleiste
- Programme zum Erstellen oder Ändern von Ikonen finden Sie als Shareware im Internet oder integriert in Entwicklungssystemen

4.17. If / Else / Endif

Funktion

Mit der `if`-Anweisung können Sie Teile des GuiXT Skripts von Bedingungen abhängig machen. Beispielweise vom aktuellen GUI-Status oder von der Existenz bestimmter Felder im SAP Bild.

Bespiel

```
if Q[Status=ADD1] or Q[Status=UPD1]
Image (1,50) (10,90) "imgupd.gif"
else
Image (1,50) (10,90) "imgdis.gif"
endif
```

Falls der GUI-Status `ADD1` oder `UPD1` vorliegt, wird die Bilddatei `"imgupd.gif"` angezeigt, sonst `"imgdis.gif"`.

Format

Sie können erstens die Existenz von Bildelementen abfragen, indem Sie einfach das betreffende Bildelement hinter `if` nennen, zum Beispiel bedeutet:

- `if F[Betrag]`: "falls ein Feld `Betrag` angezeigt wird".
- `if I[Betrag]`: "falls ein Eingabefeld `Betrag` angezeigt wird".
- `if P[Stornieren]`: "falls eine Drucktaste `Stornieren` angezeigt wird".
- `if M[Ändern]`: "falls ein Menüeintrag `Ändern` existiert".
- `if M[Ändern=active]`: "falls ein Menüeintrag `Ändern` existiert und aktiv ist".

Zweitens können Sie verschiedene Status-Informationen abfragen:

- `if Q[Status=xxxx]` Abfrage auf GUI-Status xxxx
- `if Q[Language=x]` Abfrage auf Sprachenschlüssel (einstellig, E=Englisch, D=Deutsch, F=Französisch, I=Italienisch,...)
- `if Q[Database=xxx]` Abfrage auf Systemkennung
- `if Q[Client=xxx]` Abfrage auf Mandant
- `if Q[User=xxx]` Abfrage auf Benutzer
- `if Q[Role=xxx]` Abfrage auf Rolle des Benutzers (siehe

	Rollenspezifische Benutzeroberflächen)
	• `if Q[Profile=xxx]` Abfrage auf Profile des Benutzers (siehe Rollenspezifische Benutzeroberflächen)
	• `if Q[Transaction=xxxx]` Abfrage auf Transaktionscode (Achtung: erst ab SAP Rel.4 möglich)
	• `if Q[Page=xxxx]` Abfrage auf die aktuelle Seite eines "tabbed dialog"; für xxxx setzen Sie den Text des entsprechenden Reiters ein
	• `if Q[ScreenVariant=xxxx]` Abfrage auf eine SAP Transaktionsvariante (zentrales GuiXT script)
	• `if Q[GuiXTVersion=yyyy Qx x]` Abfrage auf aktive GuiXT Version (durch <, = oder >. Siehe auch unten Tips&Tricks)
	• `if Q[StopRequest]` Abfrage in einem InputScript, ob der Benutzer einen StopRequest (siehe `StatusMessage`) ausgelöst hat.
	• `if V[vname=Wert]` Abfrage auf den Wert der Variable vname (nur möglich mit InputAssistant)
	• `if U[uname=Wert]` Abfrage auf den Wert des using-Parameters uname (nur möglich mit InputAssistant)
	
	Drittens können Sie beliebige Benutzeroptionen, die in guixt.ini als
	`Option opt1`
	`Option opt2`
	
	gesetzt sind, durch `Q[Option=xxxx]` abfragen. Beispielsweise können Sie eine Option `HelpDisplay` anbieten, bei der Sie eine auf die jeweilige Transaktion zugeschnittene Dokumentation anzeigen (als Bilddatei mit `Image` oder als rtf oder html Datei mit `View`). Jeder Benutzer kann dann durch Setzen der Option `HelpDisplay` selbst entscheiden, ob er die Erläuterungen angezeigt haben möchte oder nicht.
	
	Sie können in guixt.ini maximal 50 Optionen definieren. Jeder `Option`-String kann bis zu 30 Zeichen lang sein.

Sind Verschachtelungen von if und else möglich ?
 	
Können logische Ausdrücke eingesetzt werden ?
	Ja, Beides ist möglich. Beispiel:
	`if not Q[Status=UPD1]`
	`Image (1,50) (10,90) "imgdis.gif"`
	`else`
	`if (F[Company code] or F[Business area]) and notF[Order number]`
	`Image (1,50) (10,90) "imgupd1.gif"`
	`else`
	`Image (1,50) (10,90) "imgupd2.gif"`
	`Pushbutton (Toolbar) "Cancel" "/OZC27"`
	`endif`
	`endif`

	Sie können normale Klammern, "and", "or" und "not" innerhalb von logischen Ausdrücken einsetzen. Es gelten die normalen logischen Regeln für Reihenfolge und Klammern.

Option -strict bei Variablenvergleich	Bei dem Vergleich einer Variablen auf einen Wert durch `if V[vname=Wert]` (InputAssistant) gelten folgende Regeln:
	• Gross-/Kleinschreibung wird ignoriert
	• Ein Wert der nur aus Nullen "0000" besteht wird wie eine leere Zeichenreihe behandelt
	Sie können diese Vergleichsregeln ausser Kraft setzen, also exakt den angegebenen Wert prüfen, durch den Zusatz -strict:
	`if V[vname=Wert] -strict`
	`...`
	`endif`

Tipps & Tricks	• Die Abfrage auf einen Feldinhalt, z.B. das Feld F[Buchungskreis], ist nur über Variablen möglich:
	`Set V[buk] "&F[Buchungskreis]"`
	`if V[buk=0001]`
	`...`
	`endif`
	• Der Vergleich zweier Variablen V[x1] und V[x2] geschieht durch Einsetzen des Vergleichswertes &V[x2], also:
	`if V[x1=&V[x2]]`
	`...`
	`endif`
	• Erläuterungen zu den Abfragen `if Q[Role=xxx]` und `if Q[Profile=xxx]` finden Sie in dem Abschnitt Rollenspezifische Benutzeroberflächen
	• Die Abfrage auf die aktuelle GuiXT Version ist ab GuiXT 2002 Q4 3 unterstützt. Wollen Sie z.B. in einem Skript (meist wohl dem logon Script) sicherstellen, daß am Arbeitsplatz eines Benutzers mindestens Version 2002 Q4 5 installiert ist, so können Sie folgende Abfrage verwenden:
	`if Q[GuiXTVersion<2002 Q4 5]`
	`Message "Bitte neue GuiXT Version installieren \nBei Fragen: S. Bauer Tel 2649" Title="Update erforderlich"`
	`endif`
	In Versionen vor 2002 Q4 3 würde jedoch keine Nachricht ausgegeben, da die Abfrage dort noch nicht unterstützt war und die Bedingung Q[GuiXTVersion<....] daher als falsch angesehen wird. Wenn Sie mit älteren Versionen rechnen, fragen Sie besser mit "not ... > " und einer um eins verringerten Versionsnummer ab:

```
if not Q[GuiXTVersion>2002 Q4 4]
Message "Bitte neue GuiXT Version installieren \nBei
Fragen: S. Bauer Tel 2649" Title="Update erforderlich"
endif
```

- In einem InputScript ist es sehr verwirrend und keinesfalls zu empfehlen, `Screen`-Anweisungen innerhalb von `if ... endif` zu benutzen.
  Schlechtes Beispiel:

```
Screen S1
Set V[x] "a"
Enter

if V[x=b]

Screen S2
Enter "xx"

else

Screen S2
Enter "yy"

endif
```

  Nehmen wir an, es erscheint zunächst Screen S1, dann Screen S2. Wird nun Enter "xx" oder Enter "yy" ausgeführt? Tatsächlich wird im Beispiel oben Enter "xx" ausgeführt, da GuiXT nach Verarbeiten von Screen S1 eine passende Screen-Anweisung zu dem Screen S2 sucht, ohne eventuell noch offene if-Schachtelungen früherer Screen-Blöcke zu berücksichtigen.
  Stattdessen bitte immer die if-endif-Schachtelung pro Screen-Anweisung abschließen:

```
Screen S1
Set V[x] "a"
Enter

Screen S2
if V[x=b]
Enter "xx"
else
Enter "yy"
endif
```

  Falls nötig, können Sie auch mit goto/label arbeiten:

```
Screen S1
Set V[x] "a"
Enter

if V[x=b]
goto l_b
endif

Screen S2
Enter "xx"
goto l_weiter
```

```
label l_b
Screen S2
Enter "yy"

label l_weiter
```

4.18. IgnoreWarning

Funktion	Mit der `IgnoreWarning`-Anweisung können Sie Warnungsmeldungen der SAP-Anwendung automatisch quittieren, ohne dass sie dem Benutzer angezeigt werden.

Beispiel	`IgnoreWarning FX783`
	`IgnoreWarning FX248`
	
	Falls die Warnung FX783 oder FX248 in dem aktuellen Bild ausgegeben wird, quittiert GuiXT sie automatisch ohne dass sie dem Benutzer angezeigt wird. Mit der Option `toText=` können Sie Warnungen in einem Langtext sammeln und später anzeigen.

Format	`IgnoreWarning xxnnn` Dabei ist xxnnn die SAP-Kennung der Nachricht, bestehend aus einem Arbeitsgebiet xx und einer fortlaufenden Nummer nnn. Die Kennung wird bei Drücken von F1 auf den Nachrichtentext angezeigt.
Zusatzoptionen	`toText=` Zusätzlich werden die Nachrichtenkennung und der Nachrichtentext an die Textvariable mit dem angegebenen Namen als neue Zeile angehängt.

Beispiel	`IgnoreWarning F5000totext="warnings"`
	`IgnoreWarning F5193toText="warnings"`
	
	`// Anzeige der Warnungen`
	`if text[warnings]`
	`Message "&text[warnings]"`
	`Set text[warnings]""`
	`endif`

4.19. Image

Funktion	Mit der`Image`-Anweisung können Sie Bilddateien in den SAP Dynpros anzeigen. Einerseits kann das dem Benutzer die Orientierung im SAP erleichtern, da er schneller erkennt, wo er sich befindet; hierzu reichen einige kleine, charakteristische Bilder aus. Darüber hinaus kann man insbesondere für neue Benutzer Hintergrundinformationen zur jeweiligen Transaktion anzeigen, z.B. ein Diagramm, das den gesamten Geschäftsprozess darstellt. Diese Skripte könnten sehr gut in einem besonderen GuiXT-Verzeichnis abgelegt werden, da sie den geübten Benutzer wohl eher stören. Schließlich können durch die Verwendung von Variablen im Namen der Bilddatei auch variable Abbildungen (z.B. eine Produktabbildung) eingeblendet werden.
Beispiel	`Image (10,40) "C:\Images\logo.jpg"` Die Bilddatei`C:\Images\logo.jpg`wird ab Position`(10,40)`angezeigt, d.h. die linke obere Ecke des Bildes befindet sich auf Zeile `10`, Spalte `40`.

Bildformate

Unterstützt sind die Formate

- bmp
- gif
- jpg

Im allgemeinen ist es sinnvoll, bmp für kleinere Bilder (Ikonen z.B.) zu verwenden und größere Bilder komprimiert (gif oder jpg) abzulegen.

Format

- `Image (Zeile,Spalte)"Name der Bilddatei"`

Die Angaben `(Zeile,Spalte)` stellen die linke obere Ecke des Bildes dar. Sie können als Dezimalzahlen (mit Punkt, nicht Komma) angegeben werden, um das Bild exakt zu positionieren. Das Bild wird in Originalgröße angezeigt. Die Bilddatei wird, falls kein Laufwerksbuchstabe angegeben ist, im aktuellen GuiXT-Skript-Verzeichnis gesucht, ansonsten in dem angegebenen Laufwerk. Die Bilddateien werden nach erstmaligem Laden intern abgespeichert (Puffermechanismus).

Statt `(Zeile,Spalte)` ist auch die Angabe `(Toolbar)` unterstützt. In diesem Fall sollte das Bild eine Größe von 18x18 Pixel haben. Es wird rechts aussen in der Toolbar angezeigt.

- `Image (Zeile1,Spalte1) (Zeile2,Spalte2) "Name der Bilddatei"`

Hier ist `(Zeile1,Spalte1)` die linke obere Ecke und `(Zeile2,Spalte2)` die rechte untere Ecke. Das Bild wird horizontal und vertikal gedehnt, um exakt in den angegebenen Bereich hineinzupassen, die Proportionen des Bildes werden dabei unter Umständen verändert. Durch die Option `-NoStretch` kann man sicherstellen, daß das Bild in seinen Proportionen unverändert bleibt.

Optionen

`-NoStretch`	Das Bild wird nicht horizontal oder vertikal gestreckt, sondern behält seine ursprünglichen Proportionen bei.
`-Plain`	Der standardmäßige 3D-Rahmen um das Bild wird weggelassen.
`-NoBuffer`	Das Bild wird bei jeder neuen Anzeige neu geladen. Ohne diese Option werden Bilder von GuiXT im Hauptspeicher gepuffert und nicht neu vom Dateisystem gelesen. Sinnvoll ist diese Option, wenn durch externe Programme (z.B. aus SAP oder zeitgesteuert) jeweils aktuelle Versionen des Bildes zur Verfügung gestellt werden, die auch in aktuellem Zustand zur Anzeige gebracht werden sollen.
`-Transparent`	Anwendbar auf .gif und 8-bit .bmp Bilder. Das Bild wird transparent angezeigt. Technisch gesehen wird die Farbe der linken oberen Bildecke bei der Anzeige überall durch die Hintergrundfarbe des SAP

	Fensters ersetzt.
Multiple Abbildungen	Es ist auch möglich, durch die `Image`Anweisung mehrere Bilder blätterbar anzuzeigen. Hierzu benennt man die anzuzeigenden Bilder mit einer fortlaufenden Nummer, z.B.`"bild01.gif", "bild02.gif", "bild03.gif"`. In der `Image`Anweisung ersetzt man die fortlaufende Nummer durch das Symbol `&#`für einstellige Nummern 1,2,3..., `&##`entsprechend für zweistellige Nummern 01, 02, 03 oder `&###`für dreistellige Nummern usw. `Image (10,40) (16,65) "bild&##.gif"` Falls im Namen des Bildes eine Nummernvariable `&#`enthalten ist, so wird diese bei der ersten Anzeige des Bildes durch "1" ersetzt. Klickt der Benutzer auf das Bild, dann wird die Zahl erhöht (bei rechtem Mausklick verringert) und das entsprechende Bild geladen, falls vorhanden. Der Mauscursor wird, sobald er über einem multiplen Bild positioniert ist, in Form der Bildnummer dargestellt. Vergleichen Sie hierzu das GuiXT Tutorial 6 Abbildungen hinzufügen.
Datenabhängige Bilder	Im Namen der Bilddatei können Variablen enthalten sein, die zur Laufzeit durch die aktuellen Werte ersetzt werden. Beispiel: `Image (4,80)"C:\products\img&[Material].jpg"` Wurde beispielsweise die Materialnummer `00500187` eingegeben, so wird nun das Bild`C:\products\img00500187.jpg`angezeigt. Vergleichen Sie hierzu das GuiXT Tutorial 6 Abbildungen hinzufügen.

Dynamische Bildbeschaffung	Falls es nicht möglich oder sinnvoll ist, alle Abbildungen lokal abzulegen, können Sie durch die Angabe eines .exe- oder .bat-Files die Bilddatei auch erst bei Bedarf in das lokale Verzeichnis kopieren. Hierzu geben Sie hinter dem Namen der Bilddatei den Namen des Beschaffungsprogramms wie folgt an:
	`Image (Zeile,Spalte)"bilddatei" exe="programmname"`
	Wird die Bilddatei nicht gefunden, wird das angegebene Programm ausgeführt. Beim Start des Programms wird als erster Parameter der Name der gesuchten Bilddatei angegeben. Die Bilddatei wird vom Programm ins lokale Verzeichnis kopiert und anschließend geöffnet. Beispiel:
	`Image (10,40) (18,74)"C:\tempimg\chart&[konto]\&[Datum].gif" exe="C:\prog\getchart.exe"`

Interaktion mit SAP durch Bilder	Durch einen Mausklick auf das Bild können optional Benutzereingaben im SAP simuliert werden. Das geschieht durch den Zusatzparameter`Input="string":` `Image (Zeile,Spalte)"bilddatei" Input="string"` Sie können damit

- Eingabewerte in die aktuellen Eingabefelder des SAP-Bildes setzen
- Funktionen im SAP-System auslösen

Der String hat folgendes Format:

```
Input="feld1:wert1;feld2:wert2;feld3:wert3;OK:fcode"
```

Bei dem Funktionscode `fcode` kann es sich um einen internen Code aus dem Menü handeln, z.B. `SCMP`, oder einen Transaktionsaufruf wie z.B. `/NFB01` oder `/OMM02`.

So finden Sie die internen Codes: Wählen Sie im Menü der entsprechenden Transaktion die gewünschte Funktion mit der Maus an, drücken Sie dann (während der Mauscursor auf diese Funktion zeigt) die Taste **F1**. In einem Popup stellt das SAP System Ihnen nun den internen Funktionscode zur Verfügung.

Die **Enter-Taste**, ohne speziellen Funktionscode, kann durch ein `OK` am Ende des **URL-Strings** ausgelöst werden:

```
Input="feld1:wert1;feld2:wert2;feld3:wert3;OK"
```

Falls Sie mit GuiXT-Mitteln einen Touchscreen unterstützen, sind folgende Funktionen hilfreich zur Navigation innerhalb eins SAP-Controls, z.B. des Grid-Controls. Sie können damit die Navigationsfunktionen der Tastatur (vorwärts/rückwärts um eine Zeile oder Seite, Anfang oder Ende der Liste) jeweils durch ein eigenes Bild darstellen und lösen die Funktion durch Input=... aus. Beispiel:

```
Image (0,1) (3,8) "nextline.gif" Input="X[Grid1]:<DOWN>"
```

Folgende Funktionen stehen zur Verfügung:

- <DOWN> Pfeiltaste nach unten
- <UP> Pfeiltaste nach oben
- <LEFT> Pfeiltaste nach links
- <RIGHT> Pfeiltaste nach rechts
- <NEXT> Taste Bild vorwärts
- <PRIOR> Taste Bild rückwärts
- <END> Taste Ende
- <HOME> Taste Pos 1
- <BOTTOM> Taste Ctrl+Ende
- <TOP> Taste Ctrl+Pos 1

Start eines InputScript

In Verbindung mit InputAssistant können Sie auch ein InputScript starten. Hierzu geben Sie unter `process=` nach der Angabe `OK:fcode` den Namen des InputScript an.

	Beispiel: `Input="feld1:wert1;feld2:wert2;feld3:wert3; OK:fcode,process=displaymaterial.txt"` "`Using`" Parameters können ebenfalls mitgegeben werden durch die Notation `U[name]:wert`. Sie können auch GuiXT Variablen einen Wert zuweisen durch `V[name]:wert` und darauf im InputScript oder GuiXT Script zugreifen.
Beispiel	`Input="Material:00012345;Werk:0021;OK"` Bei Klick auf das Bild werden die Felder "`Material`"und"`Werk`" gefüllt und die **Enter-Taste** ausgelöst.
Start von PC-Anwendungen über Bilder	Durch einen Mausklick auf das Bild können optional PC Anwendungen gestartet werden. Das geschieht durch den Zusatzparameter`Start="string"`: `Image (Zeile,Spalte)"bilddatei" Start="string"` Sie können damit • Dokumente beliebigen Types zur Anzeige bringen, falls zu dem jeweiligen Dateityp in Windows ein Anzeigeprogramm registriert ist • Beliebige .exe oder .bat-Dateien mit Parametern starten
Beispiel	`... Start="\\P00800\Doc12\me21.html"` Bei Klick auf das Bild wird das html-Dokument `\\P00800\Doc12\me21.html`angezeigt.
Telefon-Anbindung	Die Weitergabe einer Telefonnummer an das TAPI-Interface ("Telephony Application Programming Interface", ausgehende Anrufe) ist durch den Parameter `dial="string"` möglich.
Beispiel	`Image (10,24)"telefon.gif" dial="&F[Telefonnummber]"` Bei Klick auf das Bild wird die angegebene Telefonnummer automatisch gewählt. Voraussetzung ist die TAPI-Installation auf dem betreffenden PC.
Anzeige eigener Hilfedateien in html-Format	*Hierfür wird die Komponente "Viewer" benötigt* Durch einen Mausklick auf das Bild wird im rechten oberen Teil des SAP-Fensters die unter `ViewHelp=`angegebene html-Datei zur Anzeige gebracht. Sie kann Interaktionen mit SAP enthalten, siehe `View`-Befehl (Setzen von Feldinhalten, Aufruf von Funktionen). Sobald der Benutzer auf ein SAP Feld klickt, verschwindet die Anzeige der Hilfedatei automatisch. `Image (Zeile,Spalte)"bilddatei" ViewHelp="filename.html"`
Beispiel	`Image (toolbar)"help.gif" ViewHelp="doc_meldung.htm"`

Optionen für `ViewHelp`	`ViewHelpWidth=`	Breite der Help-Anzeige (in Pixel). Falls keine Breite angegeben ist, wird 40% der Fensterbreite verwendet. Die Breite kann auch in Prozent angegeben werden, z.B. `ViewHelpWidth=60%`
	`ViewHelpHeight=`	Höhe der Help-Anzeige (in Pixel). Falls keine Höhe angegeben ist, wird die gesamte Fensterhöhe verwendet. Die Breite kann auch in Prozent angegeben werden, z.B. `ViewHelpHeight=50%`
	`ViewHelpPosition=`	Position (linke obere Ecke) der Help-Anzeige. Angabe in der Form (Zeile,Spalte), z.B. `ViewHelpPosition=(10,50)`, oder auch relativ zu Bildelementen, z.B. `ViewHelpPosition="[Material]+(1,30)"`. Falls eine Position, aber nicht Breite und Höhe angegeben sind, wird das gesamte Fenster ab der angegebenen Position (nach rechts/unten) zur Anzeige verwendet.
Beispiel	`Image (toolbar)"help.gif" ViewHelp="doc_meldung.htm" ViewHelpWidth="200"`	
Dynamische Texte	`Textstring=`	Ausgabe eines zusätzlichen Textes in dem Bild. In den Text können durch Einfügen von "`\n`" Zeilenvorschübe erzeugt werden, z.B. `Textstring="Erste Zeile\nZweite Zeile\nDritte Zeile"`
Optionen für `Textstring`	`Textfont=`	Gewünschter Font. Default ist "Arial" bzw. "Arial Unicode MS" in Unicode-Systemen.
	`Textcolor=`	Textfarbe. Default ist Schwarz. Mögliche Angaben: `"red"` (rot), `"blue"` (blau), `"green"` (grün), `"white"` (weiß), `"black"` (schwarz), `"yellow"` (gelb), oder `"RGB(x,y,z)"` wobei x,y,z die RGB-Werte der Farbe sind.
	`Textheight=`	Texthöhe in Pixel. Default ist `10`.
	`Textweight=`	Schriftdicke zwischen `1` (dünn) bis `9` (sehr dick). Default ist `5`.

	`Textarea=`	Teilbereich des Bildes, in den der Text ausgegeben wird, in der Form (x1,y1) oder (x1,y1,x2,y2), in Pixelkoordinaten. Default: das ganze Bild. Bei negativen Angaben wird die rechte untere Ecke als Bezugspunkt genommen. Beispiele:
		`... Textarea=(10,20) // ab Offset (10,20) von linker oberer Ecke`
		`... Textarea=(-50,-50) // ein Quadrat rechts unten von 50 Pixel`
		`... Textarea=(30,50,-30,-50) // Mitte des Bildes`

	-textcenter	Der Text wird zentriert in dem angegebenen Bildbereich dargestellt. Falls der Text Zeilenvorschübe "\n" enthält, wird er lediglich horizontal zentriert.
Beispiel	Image (1.1,0.5) (10,20) "logon.gif" textstring="&V[_database]" textheight="24"textweight="8" textcolor=""yellow"	

4.20. Include

Funktion	Mit der Include-Anweisung können Sie auf eine weitere Skriptdatei verweisen.
Beispiel	Include "radiobuttons_bukrs.txt" Die Datei "radiobuttons_bukrs.txt" wird im laufenden Skriptverzeichnis gesucht. Sie wird in der gleichen Weise interpretiert, als wäre sie unmittelbar in der aktuellen Skriptdatei enthalten.
Format	Include "includename" Die Datei "includename" wird im laufenden Skriptverzeichnis gesucht. Falls sie nicht gefunden wird, erscheint eine Syntaxfehlermeldung im GuiXT Skriptfenster, und es geht im Hauptskript weiter. Wird sie gefunden, dann werden zunächst die Zeilen der Include-Datei interpretiert, anschliessend das ursprüngliche Skript fortgesetzt. Die Datei kann auch mit einem gesamten Pfad angegeben werden, z.B. Include "\\server12\guixt\includes\r105.txt" Ebenfalls möglich ist der direkte Bezug auf Dateien im SAP Web Repository oder auf http/ftp Servern, z.B. Include "SAPWR:ZGUIXT.r105.txt" Include "http://www.synactive.com/includes/r105.txt" Ein geschachtelter Aufruf ist möglich bis zur Stufe 10.
Tipps & Tricks	• In dem Include-Namen können Variablen verwendet werden, z.B. Include "inc&V[company].txt" • Durch if...endif kann ein Include von Bedingungen abhängig gemacht werden, z.B. if Q[Transaction=IW33] Include "IW33mainscript.txt" endif

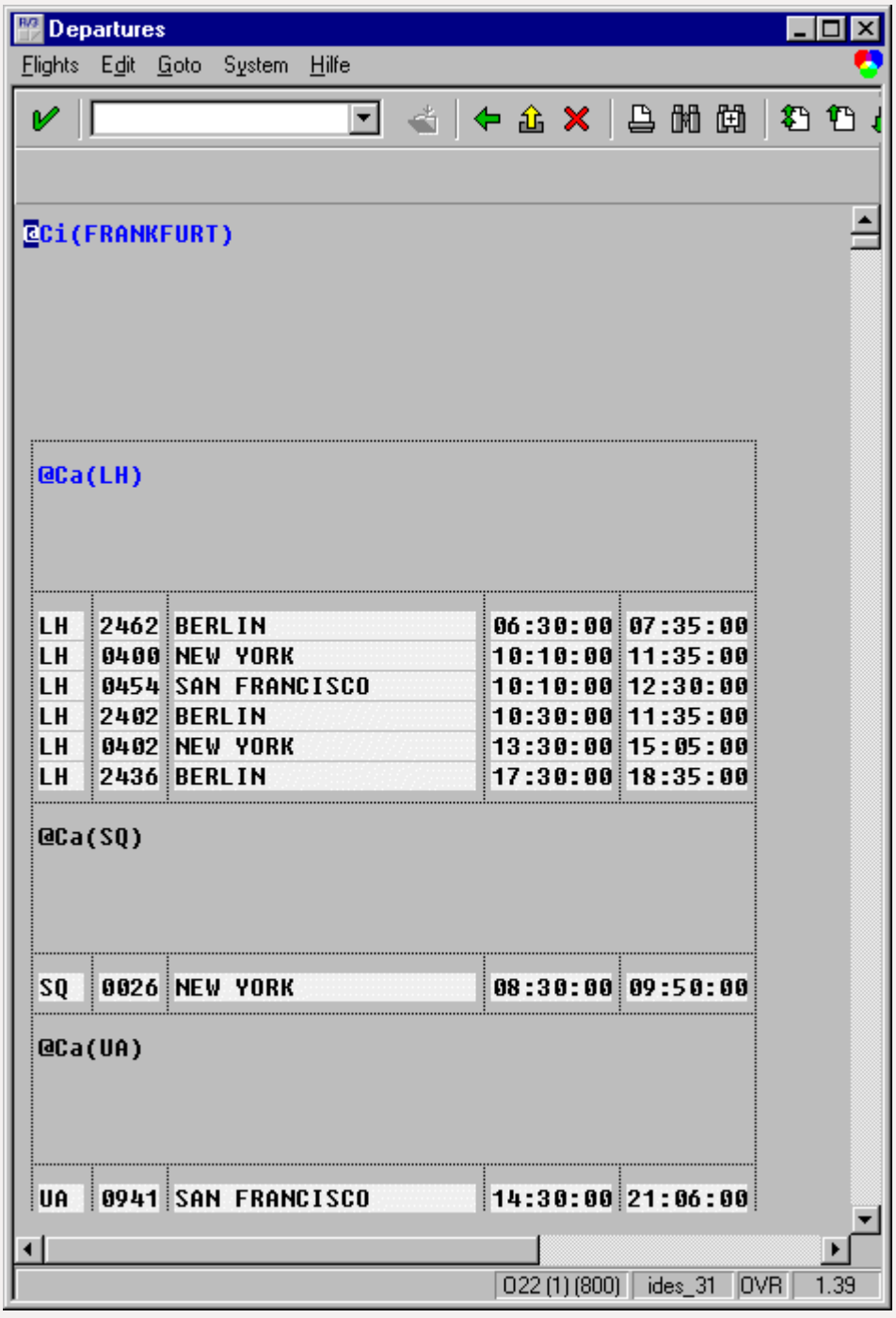
4.21. Listimage

Funktion	Mit der Listimage-Anweisung können Sie Bilddateien in den SAP ABAP Listen anzeigen. Der Funktionsumfang ist bezüglich der Bilder der gleiche wie in der Image-

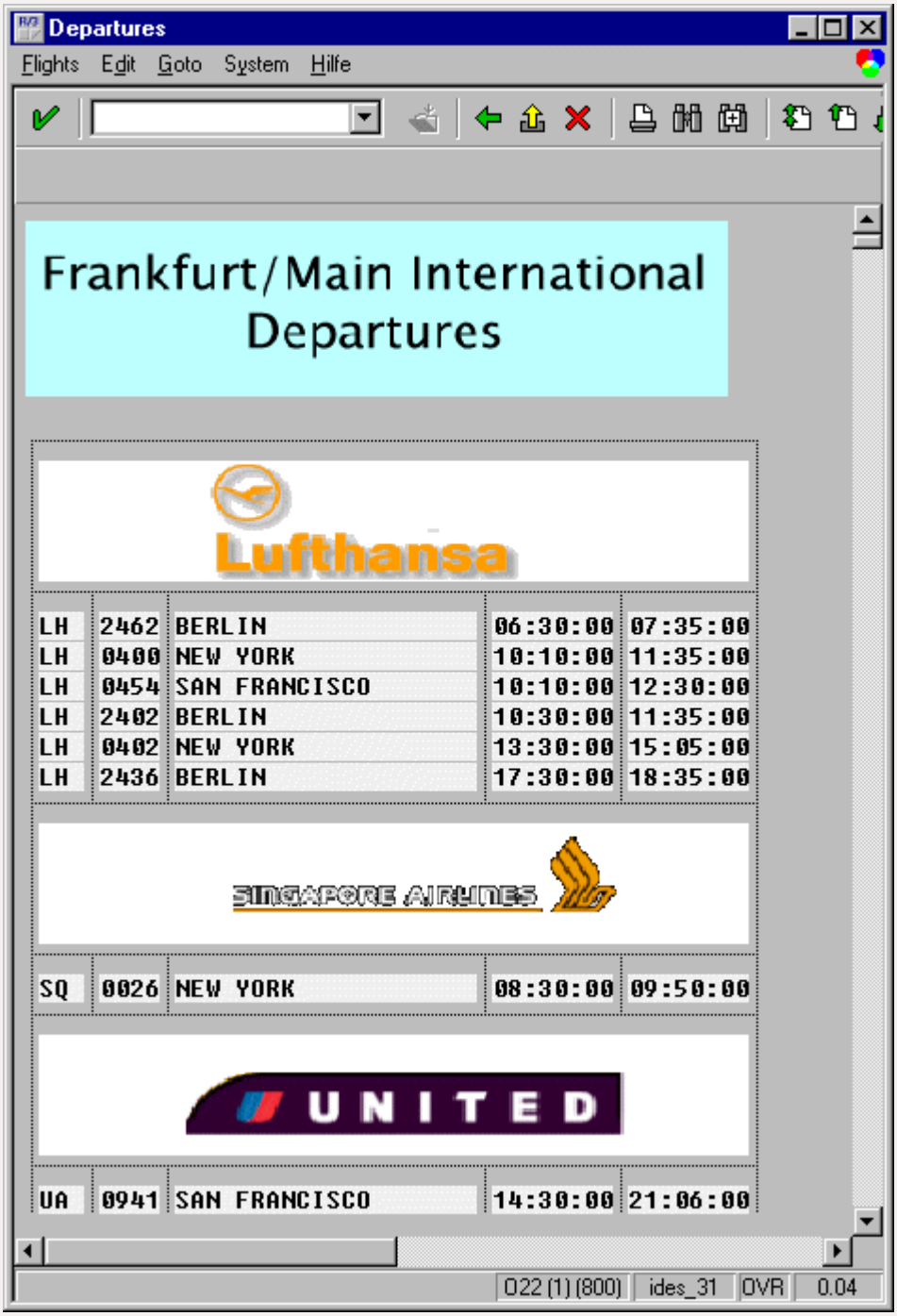
	Anweisung beschrieben. Die Position in der Liste wird durch einen frei definierbaren Suchstring festgelegt. **Einschränkungen:** • Beim Drucken der Liste werden die Bilder nicht mitgedruckt • Ein Bild wird genau dann angezeigt, wenn ohne GuiXT der entsprechende Suchstring sichtbar ist. Das heißt: Beim Weiterblättern fällt unter Umständen das Bild insgesamt weg, es wird nicht der untere Teil des Bildes angezeigt. Dagegen ist am unteren Seitenrand unter Umständen der obere Teil eines Bildes sichtbar **Bitte beachten:** Setzen Sie in Ihrem Report bitte einen Oberflächenstatus durch die ABAP-Anweisung "Set PF-Status ...". Das GuiXT-Skript kann dann unter dem von Ihnen gewählten Reportnamen angesprochen werden, also z.B. "ZIMGR000.D0120.txt". Die Dynpronummer in Listen ist immer 120. Ohne Setzen eines Oberflächenstatus wird die Liste dagegen unter dem Programmnamen "SAPMSSY0" ausgegeben und das Skript würde dann für alle Reports prozessiert, die keinen eigenen Oberflächenstatus setzen.
Beispiel	`Listimage "@Cs" (0,0) (20,50) "C:\GuiXT\Images\ffm.gif"` Die Bilddatei `C:\GuiXT\Images\ffm.gif` wird in der Liste überall statt des Strings `@Cs` angezeigt. Die Bildkoordinaten gelten dabei relativ zur Ausgabe des Suchstrings `@Cs`.

Variable Bilder	Zur Ausgabe variabler Bilder kann in der Liste hinter dem Suchstring in runden Klammern ein Wert mitgegeben werden, z.B.
	`@Cs(12345678)`
	In dem Namen der Bilddatein wird dann der String `&par` durch den angegebenen Wert ersetzt. Beispiel:
	`Listimage "@Cs" (0,0) (20,50) "\\p800\products\&par.gif"`
	Es wird die Bilddatei `\\p800\products\12345678.gif` angezeigt. In der Liste wird sowohl der Suchstring als auch der mitgegebene Wert gelöscht.

Zusatzoptionen	Es gelten die gleichen Zusatzoptionen wie in der `Image`-Anweisung; auch multiple Bilder und Interaktionen sind möglich.
Format	• `Listimage "string" (Zeile,Spalte) "Name der Bilddatei"` Die Angaben `(Zeile,Spalte)` stellen die linke obere Ecke des Bildes relativ zur Ausgabe von `"string"` dar. • `Listimage "string" (Zeile1,Spalte1) (Zeile2,Spalte2) "Name der Bilddatei"`

	Analog der `Image`-Anweisung.
Tips &Tricks	• Der Suchstring muß so eindeutig gewählt werden, daß er nicht zu unbeabsichtigter Ausgabe von Bildern führt, z.B. unter Verwendung von Sonderzeichen • Auch bei TOP-OF-PAGE oder in den festen Spaltenüberschriften einer Liste können Bilder verwendet werden
Beispiel	 `ListImage "@Ci" (0,0) (6,50) "C:\GuiXT\Images\fpar.gif" -NoStretch`

```
ListImage "@Ca" (0,0) (4,47) "C:\GuiXT\Images\&par.gif" -Plain
```



4.22. Mark

Funktion	Mit der `Mark`-Anweisung können Sie Eingabefelder durch ein vorangestelltes x markieren. Sie können z.B. in einer größeren Eingabemaske die Felder ankreuzen, die ein Benutzer ausfüllen sollte.

Beispiele	`Mark F[Gewichtseinheit]`
	`Mark F[Gewichtseinheit] -green`
	`Mark F[Gewichtseinheit] Image="C:\GuiXT\kreis.gif"`

Format	**Mark [Eingabefeld]** Das Eingabefeld wird durch ein vorangestelltes kleines rotes Kreuz markiert

Zusatzoptionen

-red	Kreuz in rot (Default)
-green	Kreuz in grün
-blue	Kreuz in blau
-yellow	Kreuz in gelb
-black	Kreuz in schwarz
-white	Kreuz in weiss
Image="filename"	Statt eines kleinen Kreuzes wird der angegebene Imagefile (z.B. .bmp, .gif) angezeigt. Das angegebene Bild sollte die Größe 8x8 Pixel haben.
-Text	Die Markierung wird vor dem Feldtext angezeigt

Anzeige eigener Hilfedateien in html-Format

Hierfür wird die Komponente "Viewer" benötigt

Durch einen Mausklick auf das Kreuz wird im rechten oberen Teil des SAP-Fensters die unter **ViewHelp=** angegebene html-Datei zur Anzeige gebracht. Sie kann Interaktionen mit SAP enthalten, siehe **View**-Befehl (Setzen von Feldinhalten, Aufruf von Funktionen).

Sobald der Benutzer auf ein SAP Feld klickt, verschwindet die Anzeige der Hilfedatei automatisch.

```
Mark [Eingabefeld] ViewHelp="filename.html"
```

Beispiel

```
Mark F[Gewichtseinheit] Image="help.gif" ViewHelp="doc_gwe.htm"
```

Optionen für **ViewHelp**

ViewHelpWidth=	Breite der Help-Anzeige (in Pixel). Falls keine Breite angegeben ist, wird 40% der Fensterbreite verwendet. Die Breite kann auch in Prozent angegeben werden, z.B. ViewHelpWidth=60%
ViewHelpHeight=	Höhe der Help-Anzeige (in Pixel). Falls keine Höhe angegeben ist, wird die gesamte Fensterhöhe verwendet. Die Breite kann auch in Prozent angegeben werden, z.B. ViewHelpHeight=50%

© by Synactive GmbH

	ViewHelpPosition=	Position (linke obere Ecke) der Help-Anzeige. Angabe in der Form (Zeile,Spalte), z.B. ViewHelpPosition=(10,50), oder auch relativ zu Bildelementen, z.B. ViewHelpPosition="F[Material]+(1,30)". Falls eine Position, aber nicht Breite und Höhe angegeben sind, wird das gesamte Fenster ab der angegebenen Position (nach rechts/unten) zur Anzeige verwendet.
Beispiel	`Mark [Gewichtseinheit] Image="help.gif" ViewHelp="doc_gwe.htm" ViewHelpWidth="200"`	
Tipps & Tricks	Falls Sie die angezeigten kleinen Kreuze an anderer Stelle benötigen, z.B. um eine Legende der verwendeten farbigen Kreuze anzuzeigen, so können Sie diese in der `Image`-Anweisung über die Namen • "mark_red.res" • "mark_green.res" • "mark.blue.res" • ... statt eines Image-Dateinamens ansprechen. statt eines Image-Dateinamens ansprechen. Vergleichen Sie auch das Viewer Tutorial 7 zur ViewHelp Komponente.	

4.23. Message

Funktion	Mit der `Message`-Anweisung können Sie aus einem GuiXT Skript oder einem InputScript heraus dem Benutzer eine besondere Nachricht anzeigen (eigenes PopUp)
Beispiel	`Message "Preisänderung ab &[AEDAT]" Title="Bitte neuen Preis beachten"`

Format	`Message "Nachrichtentext"`
	`Message "Nachrichtentext" Title="Titel"`
	Die ein- oder mehrzeilige Nachricht `"Nachrichtentext"` wird mit dem Titel `"Titel"` angezeigt.

Optionen	`-StatusLine` Die Nachricht wird in der Statuszeile angezeigt. Dabei kann unter Umständen eine Nachricht der SAP-Anwendung überdeckt werden. Falls der Nachrichtentext mit "S: " oder "E: " oder "W: " beginnt, wird die Nachricht als Information oder Fehlermeldung oder Warnung angezeigt. Beispiel: `Message "E: Betrag &V[Betrag] zu hoch, Limit ist 50.000" -`

	StatusLine • Der Title= Parameter wird in diesem Fall ignoriert • Die Option **-StatusLine** wird in "SAPGUI for HTML" Umgebung (ITS) automatisch angenommen

-yesno	Dem Benutzer stehen zwei Auswahlmöglichkeiten zur Verfügung: Ja und Nein.
	Beispiel:
	`Message "Möchten Sie Ihre Eingaben speichern?" Title="Speichern" -yesno`
	`if Q[ok]`
	`// der Benutzer hat auf den Ja-Knopf gedrückt`
	`...`
	`else`
	`// der Benutzer hat auf den Nein-Knopf gedrückt`
	`...`
	`endif`

id= **number=** **type=** **var1=** **var2=** **var3=** **var1=**	Statt der direkten Angabe eines Nachrichtentextes können Sie durch **id=** und **number=** auf eine SAP-Nachricht Bezug nehmen (SAP-Tabelle T100). Die Nachricht wird dann in der Anmeldesprache aus dem SAP-System gelesen und angezeigt. Als Nachrichtentyp type= kann "E", "W" oder "S" angegeben werden (E=Error, W=Warning, S=Success). Sie können zusätzlich bis zu vier Nachrichtenvariablen über **var1=**, ..., **var4=** mitgeben. Beispiel: `Message id="V1" number="302" var1="&V[aufnr]" type="E"-statusline` Falls im SAP-System ein Langtext zur Nachricht abgelegt ist, wird sie dem Benutzer bei Klick auf die Nachrichtenzeile angezeigt. Das Ersetzen der Variablen im Langtext ist zur Zeit nicht unterstützt. Bitte beachten: Diese Variante setzt einen gültigen RFC Benutzer im GuiXT Profile voraus. Der Nachrichtentext wird von GuiXT durch Aufruf des SAP-Bausteins BAPI_MESSAGE_GETDETAIL besorgt. Der RFC-Benutzer benötigt dazu die Berechtigung für Funktionsgruppe BATG: Berechtigung S_RFC: RFC_TYPE=FUGR, RFC_NAME=BATG, ACTVT=16
Tipps & Tricks	• Variablen können wie üblich in den Nachrichtentext oder Titel eingestreut werden • Durch **\n** können Zeilenvorschübe im Nachrichtentext eingefügt werden, durch **\t** ein Tabulator

4.24. NoAreaResize

Funktion	Mit der **NoAreaResize**-Anweisung können Sie das automatische Anpassen der Grösse interner Subscreens durch GuiXT ausschalten. Das ist nur in seltenen Fällen nötig, z.B. bei horizontal scrollbaren Teilbereichen, in denen der GuiXT-Automatismus nicht zum gewünschten Ergebnis führt.
Beispiel	**NoAreaResize**
Format	**NoAreaResize** Die Grösse aller internen Teilbereiche des aktuellen Screens bleiben unverändert auf SAP-Standard.

4.25. NoDropDownList

Funktion	Mit der **NoDropDownList**-Anweisung können ein Eingabefeld, in dem sofort eine Werteliste angezeigt wird, in ein normales Eingabefeld verwandeln.
Beispiel	**NoDropDownList F[Auftragsart]**
Format	**NoDropDownList [Bildelement]** Wandelt eine DropDownList in ein normales Eingabefeld um.
Tips &Tricks	• Sinnvoll, wenn die angezeigte Werteliste so umfangreich ist, daß eine direkte Eingabe einfacher erscheint

4.26. NoInput

Funktion	Mit der **NoInput**-Anweisung können Sie die Eingabebereitschaft eines Feldes oder einer Feldgruppe wegnehmen.
Beispiel	**NoInput F[Buchungskreis]**
Format	**NoInput [Bildelement]** **NoInput [Table,Spaltenname]** **NoInput [Table,*,Zeilennummer]** Bei Feldern, Ankreuzfeldern und Auswahlknöpfen wird die Eingabebereitschaft weggenommen. Bei Feldgruppen wird die Eingabebereitschaft aller Elemente der Gruppe weggenommen. Sie können auch eine Tabelle, eine einzelne Spalte einer Tabelle oder eine Zeile angeben.
Zusatzoptionen	**-SearchHelp** Die Eingabe über F4-Suchhilfe bleibt möglich
Beispiel	Das folgende Skript nimmt in Transaktion VA02 die Eingabe für die Spalte "Bezeichnung" sowie für alle bereits existierenden Zeilen weg (Komponente InputAssistant wird hierfür benötigt, da Variablen genutzt werden):

```
if Q[Transaction=VA02] and Q[Page=Verkauf]

  NoInput [Alle Positionen,Bezeichnung]

  // Zähler für Zeilennummern
  Set V[i] 1

  label Position

  // Position gefüllt? Dann keine Eingabe mehr
  Set V[VA02_pos] "&cell[Alle Positionen,Pos,&V[i]]"

  if V[VA02_pos]
      NoInput [Alle Positionen,*,&V[i]]
      Set V[i] &V[i] + 1
      goto Position
  endif
endif
```

Tips &Tricks

- Die Kombination von `NoInput` und `Default` ist möglich.
- Bei Angabe einer Zeilennummer bitte beachten, dass die Nummer sich auf den angezeigten Tabellenausschnitt bezieht (startet mit 1).

4.27. NoLeadingZeros

Funktion

Mit der `NoleadingZeros`-Anweisung können Sie führende Nullen bei der Ausgabe eines Felds unterdrücken. Statt z.B." 00012" wird "12" angezeigt, statt "00000" eine einzelne Null: "0".

Beispiel

```
NoleadingZeros F[Hausnummer]
```

Format

```
NoleadingZeros [Bildelement]
NoleadingZeros [Table,Spaltenname]
```

Sie können Einzelfelder oder Tabellenspalten angeben.

4.28. NoScrollbar

Funktion

Mit der `NoScrollbar`-Anweisung können Sie Blätterleisten innerhalb des Bildschirms unterdrücken.

Beispiel

```
NoScrollbar
```

Format

```
NoScrollbar
```

Alle Blätterleisten innerhalb des Bildschirms werden unterdrückt, nicht jedoch die Blätterleiste am rechten Rand. Das ist dann sinnvoll, wenn durch Löschen von Feldern ein Blättern in Bereichen innerhalb des Bildschirms unnötig geworden ist.

4.29. Numerical

Funktion

Mit der `Numerical`-Anweisung lassen Sie zur Eingabe nur Ziffern zu.

Beispiel	`Numerical F[Hausnummer]`

Format	`Numerical [Feldname]`
	`Numerical [Table,Spaltenname]`
	
	Sie können Einzelfelder oder Tabellenspalten angeben.

4.30. Offset

Funktion	Mit der `Offset`-Anweisung können Sie ein Offset in der Form `(Zeile,Spalte)` angeben, das automatisch zu allen folgenden direkt angegebenen Koordinaten addiert wird.

Beispiel	Gleichbedeutend sind:
	
	`Offset (10,21)`
	`Radiobutton (0,0) "..."`
	`Radiobutton (1,0) "..."`
	`Radiobutton (2,0) "..."`
	
	und
	
	`Radiobutton (10,21) "..."`
	`Radiobutton (11,21) "..."`
	`Radiobutton (12,21) "..."`

Format	`Offset (zeile,spalte)`
Tips &Tricks	• Bei der Positionierung einer zusammengehörenden Gruppe von Elementen, z.B.mehreren Auswahlknöpfen, ist `Offset` hilfreich • Durch `Offset (0,0)` kann man wieder auf den Ausgangszustand (kein Offset) zurückgehen • Die Anwendung von `Offset` sehen Sie auch im GuiXT Tutorial 3 und im Special Topic Wie man mit skrollbaren Bildschirmbereichen umgeht

4.31. PlaySound

Funktion	Mit der `PlaySound`-Anweisung können Sie die eine .wav-Datei abspielen.
Beispiele	`PlaySound file="intro.wav"`

Format	`PlaySound file="Dateiname"`
	
	`"Dateiname"` ist der Name einer .wav-Datei. Sie kann mit oder ohne Pfad (aktuelles Script-Verzeichnis) angegeben werden. Auch Dateien aus dem SAP Web Repository sind möglich.
	`PlaySound alias="soundalias"`
	
	`"soundalias"` ist dabei der symbolische Name eines Ereignisses, dem in der Windows

	Systemsteuerung ein Sound zugeordnet ist (Sound Schema). Beispiele: "MailBeep", "RingIn".

4.32. ProcessingOption

Funktion	Mit der`ProcessingOption`-Anweisung können Sie verschiedene Verarbeitungsoptionen dynamisch während des Ablaufs setzen.	
Beispiel	`ProcessingOption ReturnOnError="Off" ContinueOnPopup="Off"`	
Format	`ProcessingOption option="On/Off/value"` Die Option `option` wird ein- oder ausgeschaltet, oder es wird ein bestimmter Wert gesetzt. Der Default-Wert ist im Folgenden jeweils als Beispiel genannt. Die Optionen, die sich auf den Ablauf eines InputScripts beziehen, werden jeweils bei Start des InputScripts auf den Default-Wert zurückgesetzt. Alle anderen Optionen gelten pro Session und sollten im Session-Script dsession.txt (esession.txt,...) gesetzt werden.	
Optionen	`ContinueOnPopup="On"`	`On`:Falls ein Popup-Screen nicht im InputScript enthalten ist, wird das Popup angezeigt und das InputScript danach fortgesetzt. `Off`: Falls ein Popup-Screen nicht im InputScript enthalten ist, wird das InputScript mit der Anzeige des Popup-Screens beendet. Setzen in: InputScript
	`ReturnOnError="On"`	`On`:Falls die SAP-Anwendung eine Fehlermeldung ausgibt, wird die Ausgangstransaktion neu aufgerufen, die ursprünglichen Eingabewerte wieder gesetzt und die Fehlermeldung ausgegeben. `Off`: Falls die SAP-Anwendung eine Fehlermeldung ausgibt, wird das InputScript beendet und die Fehlermeldung auf dem aktuellen Bildschirm ausgegeben. Setzen in: InputScript
	`TreatWarningAsErrors="Off"`	`On`: Warnungen im Ablauf des InputScript werden wie eine Fehlermeldung behandelt. `Off`: Warnungen werden übergangen, jedoch in das Statusfenster aufgenommen (siehe `StatusMessage`). Setzen in: InputScript
	`MaxWarnings=20`	Anzahl der Warnungen, die pro Bild automatisch

	quittiert werden. Danach wird die Warnung wie eine Fehlermeldung behandelt. Default ist n=20. Setzen in: InputScript
IgnoreWarningOnNewScreen="Off"	On: Ignoriert Warnungen, falls sie bei der Anzeige eines neuen Screens ausgegeben werden. Off: Quittiert Warnungen automatisch durch "Enter" Setzen in: InputScript
IgnoreErrorOnNewScreen="Off"	On: Ignoriert Fehlermeldungen, falls sie bei der Anzeige eines neuen Screens ausgegeben werden. Off: Bricht die Verarbeitung des InputScripts ab Setzen in: InputScript
LeadingSpace="Off"	On:Bei der Übernahme von Feldinhalten durch z.B. SetV[x] " ... " werden führende Leerzeichen in dem angegebenen String ebenfalls übertragen. Off:Bei der Übernahme von Feldinhalten durch z.B. SetV[x] " ... " werden führende Leerzeichen in dem angegebenen String entfernt. Setzen in: Session-Script
TrailingSpace="Off"	On:Bei der Übernahme von Feldinhalten durch z.B. SetV[x] " ... " werden Leerzeichen am Ende des angegebenen Strings ebenfalls übertragen. Off:Bei der Übernahme von Feldinhalten durch z.B. SetV[x] " ... " werden Leerzeichen am Ende des angegebenen Strings entfernt. Setzen in: Session-Script
SearchHelpTransaction="ZXF4"	Transaktionscode, der für die Wertehilfe bei eigenen Eingabefeldern (siehe Inputfield) benutzt wird. Setzen in: Session-Script
DebugRFC="Off"	On: Jeder von GuiXT durchgeführte Remote Function Call (RFC) wird im ABAP-DEBUG-Modus ausgeführt. Das betrifft auch interne Calls (z.B. Lesen aus SAP Web Repository). Off: Normale Verarbeitung ohne Debugging.
KeepRfcSystem="Off"	On: Falls die SAP-Anwendung über RFC eine

	Transaktion in einem anderen System aufruft, werden trotzdem die Skripte des aufrufenden Systems verwendet. Der GuiXT Profile-Parameter "EnableSystem" wird für das aufgerufene System nicht wirksam. Off: Falls die SAP-Anwendung über RFC eine Transaktion in einem anderen System aufruft, werden die Skripte des aufgerufenen Systems verwendet. Der GuiXT Profile-Parameter "EnableSystem" wird für das aufgerufene System wirksam.
RfcCodepage="none"	Angabe eine SAP-Codepage, die bei allen folgenden RFC-Aufrufen verwendet wird. Falls nicht angegeben, verwendet GuiXT in Unicode-Systemen UTF-8, sonst die aktuelle Codepage der SAP-Session.
HistoryFont="System" HistoryFontSize=16 HistoryFontWeight=5	Auswahl einer Schriftart und Schriftgrösse für die Anzeige der Eingabehistorie und lokalen Wertehilfe. Weight: Schriftdicke zwischen 1 (dünn) bis 9 (sehr dick).
MaxStepcount=1000000	Maximale Anzahl Scriptzeilen, die prozessiert werden. Default ist 1 Million, dann erfolgt ein Abbruch. Setzen in: InputScript oder GuiXT Script. **Bitte beachten:** In einem InputScript setzt GuiXT "MaxStepCount" mit jeder Screen-Anweisung wieder auf den Default-Wert zurück.
RfcDisconnect=10	Halten der RFC-Verbindungen über Dialogschritte hinweg. Folgende Angaben sind möglich: On: oder 1: Nach jedem Dialogschritt werden alle RFC-Verbindungen geschlossen. Off: oder 0: Alle RFC-Verbindungen werden unbegrenzt gehalten. n>1: Eine RFC-Verbindung wird geschlossen, wenn sie in den letzten n Dialogschritten nicht benutzt wurde.
InputScriptTrace="Off" .. file="filename"	On: Schaltet den InputScript Trace ein Off: Schaltet den InputScript Trace aus

		Die Ausgabe erfolgt in die angegebene Datei; neue Sätze werden jeweils angehängt. Sie können in dem Dateinamen den String &time verwenden; er wird durch die aktuelle Uhrzeit (Stunde und Minute, Format hh.mm) ersetzt. Zum Beispiel: `ProcessingOption InputScriptTrace="On" file="C:\temp\trace.&time.txt"` Die Ausgabe erfolgt dann um 16:34h in die Datei "C:\temp\trace.16.34.txt"

Tipps & Tricks	Wenn Sie die Option `ProcessingOption DebugRFC="On"` gesetzt haben, kann es notwendig sein, sie durch `ProcessingOption DebugRFC="Off"` wieder auszuschalten, zum Beispiel:
	
	Verwenden Sie bitte
	
	`ProcessingOption DebugRFC="On"`
	`Call ...`
	`ProcessingOption DebugRFC="Off"`
	
	Das heißt, der `Call`-Befehl schaltet die Debug-Option nicht um.	

4.33. Pos

Funktion	Mit der `pos`-Anweisung können Sie Felder, Feldgruppen, Drucktasten, Auswahlknöpfe, Ankreuzfelder und Table Controls innerhalb des Bildes verschieben.

Beispiele	`pos F[Buchungskreis] (10,50)`
	
	`pos G[Weitere Daten] G[Weitere Daten]+(10,0)`
	
	`pos #[12,60](12,50)`
	
	`pos T[ORDERS](10,20)`

Format	`pos [Bildelement] (Position)`
	
	Bei Feldern wird sowohl die Feldbezeichnung als auch der dahinter stehende Wert bzw. das dahinter stehende Eingabefeld verschoben. Bei Feldgruppen werden alle Elemente innerhalb des Gruppenrahmens und der Gruppenrahmen selbst verschoben.
	
	Sie können mit `pos` auch die Position der einzelnen tabs innerhalb eines tabstrip verändern. Beispiel:
	
	`pos P[Einkauf] 1`
	`pos P[Werksbestand] 2`
	`pos P[Prognose] 3`

Optionen

`-Triple`	Sinnvoll bei Ein-/Ausgabefeldern mit einem dahinter angezeigten Klartext. Auch dieses dritte Feld wird verschoben
`-Value`	Möglich bei Ein-/Ausgabefeldern mit vorangestelltem Text. Es wird das Feld, nicht jedoch der Text verschoben
`-Text`	Nur sinnvoll bei Ein-/Ausgabefeldern mit vorangestelltem Text. Es wird lediglich der Text, nicht das Feld selbst verschoben
`width=`	Möglich bei speziellen Controls X[...], z.B. einem grid control: legt die Breite des Controls fest
`height=`	Möglich bei speziellen Controls X[...], z.B. einem grid control: legt die Höhe des Controls fest

Tipps & Tricks

- Falls man alle Felder innerhalb eines Gruppenrahmens bis auf eins verschieben möchte, kann man das auszusparende Feld zunächst durch `pos` aus dem Gruppenrahmen heraus an eine andere Position verschieben und anschließend die Feldgruppe verschieben
- Wenn in einem Skript-Kommando eine Position symbolisch durch die Position eines anderen Bildelements spezifiziert ist, und dieses Bildelement durch `pos` verschoben wird, dann gilt in allen Skript-Zeilen vor dem `pos`-Kommando die alte Position, danach die neue. Beispiel:
```
text F[Konto]+(0,50) "Text 1"
pos F[Konto] F[Konto]+(1,0)
text F[Konto]+(0,50) "Text 2"
```
Die beiden Textstrings `"Text 1"` und `"Text 2"` erscheinen dann in zwei Zeilen untereinander
- Um zwei Felder zu vertauschen, reicht es deshalb nicht aus, z.B. :
```
pos F[Konto] F[Währung]
pos F[Währung] F[Konto]
```
zu schreiben; man muß in der zweiten `pos`-Anweisung absolute Koordinaten verwenden oder sich auf ein anderes Bildelement beziehen.
- Erläuterungen zum Schlüsselwort `pos` finden Sie auch im GuiXT Tutorial 1 Layout ändern

4.34. Pushbutton

Funktion

Mit der `Pushbutton`-Anweisung können Sie eigene Drucktasten innerhalb eines SAP-Bildes oder in der Drucktastenleiste definieren. Sie können sowohl Funktionen aus dem Menü als Drucktaste direkt erreichbar machen, oder die Navigation zu anderen Transaktionen auf eine Drucktaste legen.

Beispiel

```
Pushbutton (10,50) "Split Screen Editor" "SCMP"
```

Es wird auf Zeile `10`, Spalte `50` eine Drucktaste mit dem Text `Split Screen Editor` hinzugefügt. Ein Mausklick auf die Drucktaste löst den internen Code `SCMP` aus, der (im Beispiel SE38) zum Aufruf des Split Screen Editors führt.

Interne Codes

Wählen Sie im Menü der entsprechenden Transaktion die gewünschte Funktion mit der Maus an, drücken Sie dann (während der Mauscursor auf diese Funktion zeigt) die Taste **F1**. In einem Popup stellt das SAP System Ihnen nun den internen Funktionscode zur Verfügung.

Format

- `Pushbutton (Zeile,Spalte)"Text auf Pushbutton" "FCode" Process="..."`
  Drucktaste im Bild auf Position `(Zeile,Spalte)`. Die Drucktaste löst den Code `FCode` aus. Dabei kann es sich um einen internen Code aus dem Menü handeln, z.B. `SCMP`, einen Transaktionsaufruf wie z.B. `/NFB01` oder `/OMM02` oder eine F-Taste oder eine Kombination von F-Tasten im Format /nn.
  `Process="..."` verweist auf eine Datei, in der das InputScript abgelegt ist. Die "fcode"-Angabe kann entfallen, falls nur "Enter" ausgelöst werden soll.
- `Pushbutton (Toolbar) "Text auf Pushbutton" "FCode"` Drucktaste in der Drucktastenleiste. Hierfür wird automatisch eine freie F-Taste vergeben und in der Quickinfo angezeigt.
- `Pushbutton (Toolbar) "Text auf Pushbutton" "FCode" "Fkey"` Wahlweise kann man auch über einen weiteren Parameter die F-Taste mitgeben, und zwar im Format **F1,..F12, Shift+F1,...,Shift+F12, Ctrl+F1,...,Ctrl+F12, Shift+Ctrl+F1, ..., Shift+Ctrl+F12**. Falls diese Funktionstaste bereits belegt ist, wird automatisch eine andere, freie Taste zugeordnet
  z.B.: Shift+F5 -> F17, F3 -> F3

Ikonen und Quickinfo

Der Drucktastentext kann durch eine SAP Ikone und eine Quickinfo ergänzt werden. Das geschieht in der SAP-Konvention

```
"@id\Qquickinfo@text"
```

Hier ist `id` das Kennzeichen der SAP-Ikone, `quickinfo` der als Quickinfo (Tooltip) anzuzeigende String, `text` der Drucktastentext. Sie finden eine Liste aller SAP Ikonen in der Transaktion ICON, leider dort ohne die internen Kennzeichen der Ikonen. Diese sind in dem ABAP Programm <ICON> abgelegt (Transaktion SE38, Programmname <ICON> eingeben, Quelltext anzeigen). Vergleichen Sie hierzu auch den Tip 24 Choosing an icon.

Beispiel:

```
Pushbutton (10,50) "@3G\QAufruf des Split Screen Editors@Split Screen" "SCMP"
```

Zusatzoptionen

`Size=x`	Die Drucktaste erstreckt sich über x Zeilen. Default ist 1, Maximum ist

	99.
`Size=(x,y)`	Die Drucktaste erstreckt sich über x Zeilen und y Spalten.
`-disabled`	Die Drucktaste ist nicht aktiv.
`-separator`	Nur in Verbindung mit `(Toolbar)`. Die Drucktaste wird durch eine kleine senkrechte Linie von den links stehenden Drucktasten abgesetzt.

Tipps & Tricks

- Sie können Drucktasten aus der Drucktastenleiste löschen mit `del[Drucktastentext].`
- Zu existierenden Drucktasten können Sie durch `Tip[Drucktastentext]"Quickinfo"` eine Quickinfo hinzufügen.
- Neue Drucktasten in der Drucktastenleiste werden rechts angehängt. Sie erscheinen auch in der Übersicht über die Funktionstastenbelegung (rechte Maustaste); dort werden sie gemäß den vergebenen Funktionstasten einsortiert.
- Eine Einführung ins Thema finden Sie auch im GuiXT Tutorial 2 Eigene Drucktasten.
- Kombinationen von Funktionstasten /nn mit Ctrl und Shift können als Zahlen zwischen 13 und 48 eingegeben werden. Dabei entspricht

  Shift = +12
  Ctrl = +24

  Die Enter-Taste kann durch **/0** dargestellt werden.

  Shift+F1 /13 Ctrlt+F1 /25 Shift+Ctrl+F1 /37
  Shift+F2 /14 Ctrlt+F2 /26 Shift+Ctrl+F2 /38
  Shift+F3 /15 Ctrlt+F3 /27 Shift+Ctrl+F3 /39
  Shift+F4 /16 Ctrlt+F4 /28 Shift+Ctrl+F4 /40
  Shift+F5 /17 Ctrlt+F5 /29 Shift+Ctrl+F5 /41
  Shift+F6 /18 Ctrlt+F6 /30 Shift+Ctrl+F6 /42
  Shift+F7 /19 Ctrlt+F7 /31 Shift+Ctrl+F7 /43
  Shift+F8 /20 Ctrlt+F8 /32 Shift+Ctrl+F8 /44
  Shift+F9 /21 Ctrlt+F9 /33 Shift+Ctrl+F9 /45
  Shift+F10 /22 Ctrlt+F10 /34 Shift+Ctrl+F10 /46
  Shift+F11 /23 Ctrlt+F11 /35 Shift+Ctrl+F11 /47
  Shift+F12 /24 Ctrlt+F12 /36 Shift+Ctrl+F12 /48

  Beispiel:

  ```
  Pushbutton (toolbar) "Anhang anzeigen" "/34" "F8"
  ```

  Durch Drücken der Drucktaste wird also Ctrl+F10 ausgelöst. Der Benutzer

	könnte auch anstelle durch Drücken von "F8" die Funktion direkt auslösen .

4.35. Radiobutton

Funktion

Mit der `Radiobutton`-Anweisung können Sie ein Eingabefeld ersetzen durch eine Reihe von Auswahlknöpfen. Der Benutzer kann dann statt der Eingabe eines Wertes (zum Beispiel der Auftragsart) einfach auf einen der Auswahlknöpfe klicken. Damit wird dem ungeübten Benutzer die Bedienung der Transaktion wesentlich erleichtert. Aber auch für einen geübten Benutzer ist bei einer kleinen Auswahl an Werten, die er auf einen Blick überschaut, die Auswahl über Radiobuttons schneller als die Eingabe des Wertes.

Statt eines einzelnen Feldes können Sie auch bis zu sechs Felder und Feldinhalte pro Radiobutton-Anweisung angeben. Der Auswahlknopf symbolisiert dann die angegebene Kombination von Werten, sowohl bei der Anzeige des Bildes als auch bei der Eingabe.

Bitte beachten: zusätzlich gibt es eine `RadioButton` Syntax zum Arbeiten mit internen Variablen in InputScripts, siehe InputAssistant Dokumentation Radiobutton.

Bespiel

```
Radiobutton        (10,1)        "Barverkauf"        F[Auftragsart]        "BV"
Radiobutton       (11,1)       "Terminauftrag"       F[Auftragsart]       "LO"
Radiobutton        (12,1)         "Retoure"         F[Auftragsart]         "RE"
del F[Auftragsart]
```

Es wird auf Zeile `10`, Spalte `1` ein Auswahlknopf mit dem Text `Barverkauf`hinzugefügt. Ein Mausklick auf den Text oder Auswahlknopf ist gleichbedeutend mit der Eingabe des Wertes `"BV"`in das Feld `F[Auftragsart]`. Ebenso wird bei der Anzeige des Bildes der Wert `"BV"` im Feld `F[Auftragsart]`dadurch dargestellt, daß der Auswahlknopf `"Barverkauf"`aktiviert ist. Mit `del F[Auftragsart]` wird das Eingabefeld `F[Auftragsart]` ausgeblendet.

Format

```
Radiobutton (Position) "Text neben Radiobutton" [Feld] "Wert"

Radiobutton (Position) "Text neben Radiobutton" [F1] "W1" [F2] "W2" [F3]
"W3" [F4] "W4"
```

Bespiel

```
Radiobutton (10,50) "Nahrungsmittlel" F[Sparte] "05" F[Verkaufsbüro] "0001"
Radiobutton  (11,50)  "Kosmetik"  F[Sparte]  "14"  F[Verkaufsbüro]  "0002"
del                                                                F[Sparte]
del F[Verkaufsbüro]
```

Tipps & Tricks

- Es ist auch möglich, Auswahlknöpfe mit einem sichtbaren Eingabefeld zu kombinieren. Das kann dann sinnvoll sein, wenn zwar nicht alle Werte vorhergesehen werden können, eine kleine Teilmenge aber sehr häufig verwendet wird. Wird dann ein Auswahlknopf gedrückt, so wird dessen Wert übernommen. Wird im Feld ein Wert eingegeben, so wird dieser Wert übernommen. Wird beides in sich widersprechender Weise getan, dann zieht die Auswahl über Radiobutton.
- Falls das Bezugsfeld (im Beispiel `F[Auftragsart]`) einen Wert enthält, der keinem

Auswahlknopf entspricht, so wird kein einziger Auswahlknopf aktiviert. In der Regel sollte das Bezugsfeld nur dann aus dem Bild gelöscht werden, wenn man sicher ist, alle möglichen Werte durch die Auswahlknöpfe abgebildet zu haben.

- Radiobuttons behandeln " " und "00" gleich. Ebenfalls wird Groß-/Kleinschreibung ignoriert. Der Grund dafür ist, daß 00 in verschiedenen SAP-Masken als Leerzeichen angezeigt wird (unterschiedlich in Anzeige/Änderungsmodus) und Kleinbuchstaben in Großbuchstaben umgesetzt werden. Durch Voranstellen von "==" im Wert wird diese Automatik ausgeschaltet. Z.B.:

```
Radiobutton (7,10) "Standard" F[Abschlagszyklen] "== "
Radiobutton (7,35) "Keine Abschläge" F[Abschlagszyklen] "==00"
```

- Falls nach einer Fehlermeldung nur ein Teil der angegebenen Felder eingabebereit ist, sind diejenigen Auswahlknöpfe änderbar, für die alle angegebenen Felder eingabebereit sind oder bereits den genannten Wert enthalten.
- Ziehen Sie ggf. auch das GuiXT Tutorial 3 Unterstützen der Dateneingabe zu Rate.

4.36. Required

Funktion	Mit der `Required`-Anweisung können Sie ein Eingabefeld als obligatorische Eingabe darstellen.
Beispiel	`Required F[Alte Materialnummer]`
Format	`Required [Eingabefeld]` Das angegebene Eingabefeld wird als obligatorische Eingabe visualisiert. Eine Prüfung, ob tatsächlich etwas eingegeben wurde, findet nicht automatisch statt. Sie kann in einem InputScript erfolgen.

4.37. Stop

Funktion	Mit der `Stop`-Anweisung können Sie die Interpretation eines Skripts beenden.	
Beispiel	`Stop`	
Format	`Stop`	
Zusatzoptionen	`Stop script`	Innerhalb einer Include-Datei eines GuiXT-Skripts: Beendet ebenfalls die Verarbeitung des Hauptskripts. Ohne die Option "script" beendet "Stop" nur die Verarbeitung der Include-Datei.
Tips &Tricks	Während des Tests eines Skripts ist die `Stop` Anweisung nützlich: Sie können eine zweite Version des Skripts in der gleichen Datei stehenlassen und eine `Stop`-Anweisung davorschreiben.	

4.38. TableWidth

Funktion	Mit der `TableWidth`-Anweisung können Sie die Breite einer Tabelle in der Anzeige ändern. Sie können die Tabelle sowohl schmaler als auch breiter anzeigen.
Beispiel	`TableWidth T[Alle Positionen] 64` Die Tabelle "Alle Positionen" wird in insgesamt 64 Spalten angezeigt.
Format	`TableWidth T[Tabelle] n`

4.39. Text

Funktion	Mit der `Text`-Anweisung können Sie: • einen beliebigen Text anzeigen • eine Feldbezeichnung ändern • die Bezeichnung einer Drucktaste, eines Auswahlknopfes oder eines Ankreuzfeldes ändern

Beispiele	`Text [Buchungskreis] "Landesgesellschaft"`
	`Text (10,50) "Bitte benutzen Sie zum Drucken"`
	`Text (11,50) "das Formular M10A"`

Zusatzoptionen	`-border` — Der Text wird mit einer Umrandung (wie Ausgabefelder) angezeigt. `-intensified` — Der Text wird farblich hervorgehoben `-fixedfont` — Der Text wird in einem Font mit fester Zeichenlänge angezeigt `-label` — Der Text wird im Stil einer Feldbezeichnung angezeigt `Size="n"` — Der Text wird in der Länge `n` angezeigt (mit Leerzeichen aufgefüllt).
Format	`Text [Bildelement] "text"` `Text (Position) "text"`
Tipps & Tricks	• Falls im gesamten System eine Feldbezeichnung ersetzt werden soll, können Sie auch die Möglichkeiten des SAP Repository nutzen oder den Befehl `GlobalTextReplace`. • Zum Einfügen von optisch ansprechend gestalteten Hinweisen können Sie die `Image`-Anweisung nutzen , oder Sie können mit der `View`-Anweisung rtf-Texte (aus Word z.B. erzeugbar) oder html-Dateien im SAP-Bild anzeigen. • Wenn Sie einen Text über ein Bild legen wollen, dann verwenden Sie den Befehl

	`Image` mit der Option `Textstring=` .

4.40. TextReplace

Funktion	Mit der`TextReplace`-Anweisung können Sie Texte bei der Anzeige des SAP Bildes ersetzen.
Beispiel	`TextReplace "Buchungskreis""Landesgesellschaft"`
Format	`TextReplace "alt""neu"` In folgenden Bildelementen wird der String `"alt"` durch `"neu"` ersetzt: • Feldbezeichnungen • Bildtitel • Rahmentitel • Drucktastentexte und Quickinfo • Nachrichtenzeile • Menütexte • Spaltenüberschriften in Tabellen • Tabellentitel Die Ersetzung wird nach folgenden Regeln durchgeführt: 1. Groß/Kleinschreibung wird ignoriert, jedoch wird der erste Buchstabe im neuen Text groß geschrieben, wenn er im alten Text ebenfalls groß geschrieben war 2. Längere Texte werden vor kürzeren Texten ersetzt

Tipps & Tricks	• Die Anweisung `GlobalTextReplace` leistet die gleiche Textersetzung für alle SAP Masken
	• Im Deutschen sind die unterschiedlichen Wortendungen ("Buchungskreis**e**", aber "Landesgesellschaft**en**") manchmal ein Problem. Sie können sich helfen durch weitere `TextReplace`-Anweisungen, z.B.
	`TextReplace "Buchungskreise""Landesgesellschaften"`
	oder durch `TextReplace` Angaben für den ganzen Text, z.B.
	`TextReplace "Auswahl des Buchungskreises" "Auswahl der Landesgesellschaft"`
	• Beachten Sie bitte auch das Special Topic Unternehmensspezifische Begriffe verwenden.

4.41. Tip

Funktion	Mit der `Tip`-Anweisung können Sie zu Feldern, Feldgruppen und Drucktasten eine einzeilige Quickinfo (auch Tooltip genannt) erfassen. Sobald der Benutzer den Mauscursor kurz auf die Feldbezeichnung positioniert, erscheint darunter ein kleines gelbes Kästchen mit dem von Ihnen erfassten Text. Bei Feldgruppen wird durch `Tip`, in die rechte obere Ecke des Rahmens, ein Symbol eingeblendet, das zur Anzeige Ihrer Quickinfo dient.

Beispiel	`Tip [Vorname]"Vorname, z.B. Karl oder Herbert, auch generische Suche Her* ist möglich"`
	
	`Tip [Vertrag]"Anzeige der Vertragsdaten und der Preisvereinbarungen zum Kunden"`

Format	`Tip [Bildelement] "Quickinfo"` Bildelemente können einzelne Felder, Feldgruppen (Rahmentext) und Drucktasten sein. Bei Feldern wird ein Symbol vor den Feldtext eingeblendet, bei Feldgruppen ein Symbol in der oberen rechten Rahmenecke.
Tipps & Tricks	• Bei Eingabefeldern helfen dem Benutzer einige typische Eingabewerte oft mehr als abstrakte Erklärungen • Die verwendete Ikone kann in guixt.ini definiert werden. Sie können irgendeine der SAP Standardikonen auswählen. Die Profile-Parameter in guixt.ini lauten: `icon_tip_field` bei Feldern (Default ist "`icon_tip_field0L`") sowie `icon_tip_box` bei Feldgruppen (Default ist "`icon_tip_box35`") • Eine Darstellung aller SAP Ikonen finden Sie in Transaktion ICON, die internen Kurzbezeichnungen der Ikonen in dem ABAP Programm <ICON>. Vergleiche auch den Tip 24 Choosing an icon . • Ab einem SAP GUI 4.0 Stand können Sie auch durch "`icon_tip_fieldSpace`" in guixt.ini die Anzeige der `Tip`-Ikone ganz unterdrücken. Allerdings ist dem Benutzer dann nicht mehr ersichtlich, bei welchen Feldbezeichnern er einen Tip abrufen kann.

4.42. Title

Funktion	Mit der `Title`-Anweisung können Sie einem SAP Bild einen eigenen Titel geben. Auch in einem InputScript können Sie `Title` verwenden. Der Titel wird dann solange angezeigt, bis das InputScript beendet ist oder eine neue Title-Anweisung erfolgt. Siehe Tips&Tricks 23.
Beispiel	`Title "Festwerte pflegen für Benutzer &[Benutzer]"`
Format	`Title "text"` Die maximale Länge des Titels beträgt 70 Zeichen.
Tipps	• Beachten Sie, daß das gleiche Bild unter Umständen zum Anzeigen, Ändern und

&Tricks	Hinzufügen verwendet wird. Entweder formulieren Sie den Titel dann neutral, zum Beispiel `Title "Festwerte für Benutzer &[Benutzer]"` oder Sie führen anhand des GUI-Status mit `if` eine Fallunterscheidung durch. • Ein sinnvolles Anwendungsbeispiel von `Title` finden Sie im Tip 23 Changing the screen title during InputScript processing.

4.43. TitlePrefix

Funktion	Mit der `TitlePrefix`-Anweisung können Sie dem Titel sämtlicher SAP Bildschirme einen eigenen Teil voranstellen, z.B. die Systemkennung oder den Mandanten. Die Anweisung ist für den aktuellen Bildschirm und alle folgenden gültig. Falls das SAP Window minimiert ist, wird der Titel auch in der Taskleiste modifiziert angezeigt. Damit kann der Benutzer z.B. leicht zwischen mehreren Systemen unterscheiden.
Beispiel	`TitlePrefix "&[_database]:"` Für ein System mit der Kennung TST beginnen dann alle Titel mit "TST:".
Format	`TitlePrefix "text"`

Tips &Tricks	• Es ist sinnvoll, die `TitlePrefix` Anweisung in das **Session-Script** DSession.txt zu legen
	• Sie können die Variablen `&[_database]` und `&[_client]` verwenden.
	• Ebenfalls möglich ist die Kombination mit einer Abfrage, z.B.
	`if V[_database=TST]`
	`  TitlePrefix "Training:"`
	`endif`
	• Falls in einem Script ein eigener Titel durch `Title` gesetzt wird, so wird dieser angezeigt

4.44. TitleSuffix

Funktion	Mit der `TitleSuffix`-Anweisung können Sie dem Titel sämtlicher SAP Bildschirme einen eigenen Teil anfügen, z.B. die Systemkennung oder den Mandanten. Die Anweisung ist für den aktuellen Bildschirm und alle folgenden gültig. Falls das SAP Window minimiert ist, wird der Titel auch in der Taskleiste modifiziert angezeigt. Damit kann der Benutzer z.B. leicht zwischen mehreren Systemen unterscheiden.
Beispiel	`TitleSuffix "(&[_database])"` Für ein System mit der Kennung TST wird dann an alle Titel "(TST)" angefügt.
Format	`TitleSuffix "text"`

Tips &Tricks	• Es ist sinnvoll, die `TitleSuffix` Anweisung in das **Session-Script** DSession.txt zu legen
	• Sie können die Variablen `&[_database]` und `&[_client]` verwenden.
	• Ebenfalls möglich ist die Kombination mit einer Abfrage, z.B.
	`if V[_database=TST]`
	`  TitleSuffix "(Training)"`
	`endif`
	• Falls in einem Script ein eigener Titel durch `Title` gesetzt wird, so wird dieser angezeigt.
	• Um die jeweilige Transaktion im Titel anzuzeigen, können Sie im session-Script folgendes Coding benutzen:
	`Set V[x] "[_transaction]"`
	`TitleSuffix "(&&V[x])"`

4.45. Uppercase

Funktion	Mit der `Uppercase`-Anweisung können Sie die Eingabe sofort in Grossbuchstaben umsetzen.
Beispiel	`Uppercase [Sortfeld]`

Format	`Uppercase [Sortfeld]`
	`Uppercase [Table,Spaltenname]`
	
	Sie können Einzelfelder oder Tabellenspalten angeben.

4.46. VersionNumber

Funktion	Mit der `VersionNumber`-Anweisung können Sie bei Verwendung von Scripts aus dem SAP Web Repository oder von ftp/http Servern die Netzbelastung auf einen einzigen Server-Zugriff pro SAP-Logon reduzieren. Sie setzen in dem Skript dsession.txt (d=Sprachenschlüssel, esession.txt für Englisch) durch `VersionNumber xxx` eine frei vergebbare Versionsnummer. Das logon-Script sowie session-Skript wird immer neu aus dem Web Repository bzw. von dem angegebenen ftp/http Server geladen, nie aus dem lokalen Cache. Alle weiteren Skripte sucht GuiXT nun direkt in dem lokalen Cache, dessen Verzeichnisname aus dem GuiXT Profile entnommen und um die angegebene Versionsnummer ergänzt wird. Es fallen in der Regel dann keinerlei Zugriffe auf den Skriptserver mehr an, falls der Benutzer keine neuen SAP Bildschirme aufruft. Ohne `VersionNumber` wird dagegen pro SAP Session jedes Skript zunächst einmal neu in den Cache eingelesen. Bei 400 Benutzern, 30 verschiedenen SAP Bildschirmen pro Benutzer und 500 Dialogschritten pro Benutzer ergeben sich in einem eingeschwungenen System folgende Server-Zugriffe:

	• ohne lokalen Cache: 400*500 = 200.000 • mit lokalem Cache: 400*30 = 12.000 • mit lokalem Cache und Versionsnummer:400 * 2 = 800 Wenn Sie später eine Änderung in den GuiXT Skripten durchführen und diese bei allen Benutzern wirksam werden soll, erhöhen Sie die Versionsnummer. Bei der nächsten SAP Anmeldung liest GuiXT die neue Versionnummer, löscht das alte Cache-Verzeichnis und legt unter der neuen Versionsnummer einen neuen Cache an, der dann sukzessiv gefüllt wird.
Beispiel	`VersionNumber 0010`
Format	`VersionNumber string` Bitte achten Sie darauf, daß `string` nur Zeichen enthält, die als Verzeichnisnamen unter Windows zulässig sind, am besten nur Buchstaben und Zahlen.
Zusatzoptionen	`systemid=` — Falls mehrere SAP-Systeme mit dem gleichen Systemnamen exstieren, z.B. P70, können Sie durch systemid= einen abweichenden beliebigen Namen angeben, der statt des Systemnamens benutzt wird, um den Namen des Cache-Verzeichnisses zu bilden.
Tips & Tricks	Wie das Schlüsselwortes `VersionNumber` eingesetzt wird, um die Performance zu steigern, ist im Tip 7 Improve performance and decrease net traffic beschrieben.

4.47. View

Funktion	Mit der `View`-Anweisung können Sie Dateien der Formate html und rtf (Rich Text Format, z.B. aus Microsoft Word heraus erzeugbar) in den SAP Masken anzeigen. Die html-Anzeige bietet darüber hinaus die Möglichkeit der Interaktion mit SAP (Setzen von Werten und Auslösen von Funktionen). Falls Sie in der View-Anweisung eine template-Datei (rtf oder html) angeben, dann werden dort alle GuiXT-Variablen der Form `&[...]` durch die aktuellen Inhalte ersetzt. Die `View`-Anweisung unterstützt wie `Image` die Bildformate bmp, gif und jpg. Im Unterschied zur `Image`-Anweisung werden die Bilder dann, ebenso wie rtf und html Dokumente, in einem scrollbaren eigenen Bereich angezeigt. Die direkte Unterstützung weiterer Multimediaformate ist geplant, sie kann zur Zeit auch durch Einbettung in ein html Dokument realisiert werden (Video und Audio Dateien). Zu beachten: Die `View`-Anweisung kann nur benutzt werden, wenn der GuiXT Viewer (SAP add-on von Synactive) sowie Microsoft Explorer 4.0 installiert sind.

Beispiel

```
View (10.5,40.2) (20.5,80) "C:\html\menupr00.html"
```

Die html Datei `C:\html\menupr00.html` wird in dem Rechteck ab `(10.5,40.2)` bis `(20.5,80)` angezeigt.

Dateiformate

Unterstützt sind die Formate

- rtf
- html
- bmp
- gif
- jpg

Format

```
View (Zeile1,Spalte1) (Zeile2,Spalte2) "Name der Datei"
View (Zeile1,Spalte1) (Zeile2,Spalte2) "Name der Datei"
template="template-Datei"
View (Zeile1,Spalte1) (Zeile2,Spalte2) template="template-Datei"
```

Die Angaben `(Zeile,Spalte1)` und `(Zeile2,Spalte2)` stellen die linke obere Ecke und die rechte untere Ecke des Anzeigebereichs dar. Sie können als Dezimalzahlen (mit Punkt, nicht Komma) angegeben werden, um das Bild exakt zu positionieren. Die Datei wird, falls kein Laufwerksbuchstabe angegeben ist, im aktuellen GuiXT-Skript-Verzeichnis gesucht, ansonsten in dem angegebenen Laufwerk.

Falls Sie eine `template` Datei angeben, dann wird diese in die erstgenannte Datei kopiert; dabei werden alle GuiXT Variablen `&[....]` durch die aktuellen Werte ersetzt. Falls Sie lediglich die `template` Datei angeben (drittes `View` Format), dann kopiert GuiXT die Datei unter Ersetzung aller Variablen in eine temporäre Datei und zeigt diese an.

Zusatzoptionen

`-floating`	Zur Anzeige wird automatisch ein neues Fenster geöffnet, das unabhängig von dem SAP Bild verschoben und in der Größe verändert werden kann.
`-maximize`	In Verbindung mit -floating: Das Fenster wird maximiert angezeigt.
`-topmost`	In Verbindung mit -floating: Das Fenster wird über aller anderen Fenstern angezeigt (auch bei Klick auf anderen Fenstern).
`ReturnWindow=`	Angabe eines Variablennamens, in die die interne Nummer des neuen Fensters gestellt wird. Zu beachten: Nur möglich, wenn `View` in einem InputScript aufgerufen wird.
`CloseWindow=`	Angabe einer durch `ReturnWindow=` davor besorgten

	Fensternummer. Das Fenster wird geschlossen. Beispiel eines InputScript: Parameter CLOSE if U[CLOSE=X] View CloseWindow="&V[viewwindow]" return endif View (6,1) (30,120) template="form2.html" -floating ReturnWindow="viewwindow"
Datenabhängige Anzeige	Im Namen der Datei können Variablen enthalten sein, die zur Laufzeit durch die aktuellen Werte ersetzt werden. Beispiel: `View (4,80) (12,120) "C:\products\img&[Materialnummer].rtf"` Wurde beispielsweise die Materialnummer 00500187 eingegeben, so wird nun die Datei `C:\products\img00500187.rtf`angezeigt.
Interaktion mit dem SAP Bild	Bei der Anzeige von html Dokumenten kann über URLs im Dateisystem, Intranet oder Internet weiternavigiert werden (normale Browser-Funktionalität). Die "Zurück"-Navigation kann dabei über die rechte Maustaste erfolgen. Daneben gibt es die Möglichkeit, durch spezielle URLs • Eingabewerte in die aktuellen Eingabefelder des SAP-Bilds zu setzen • Funktionen im SAP-System auszulösen Die URL muß hierzu mit "SAP://" beginnen. Anschließend geben Sie in einem String die Feldinhalte und gegebenenfalls eine Funktion mit, und zwar im Format <A href="SAP://feld1:wert1;feld2:wert2;feld3:wert3;OK:fcode"> Da eine URL keine Leerzeichen enthalten darf, werden die Feldnamen, falls sie aus mehreren Worten bestehen, aneinander geschrieben, zum Beispiel `"AlteMaterialnummer:00012345"` statt `"Alte Materialnummer:00012345"`. Bei dem Funktionscode fcode kann es sich um einen internen Code aus dem Menü handeln, z.B. SCMP, oder einen Transaktionsaufruf wie z.B. /NFB01 oder /OMM02. Tip: So finden Sie die internen Codes: Wählen Sie im Menü der entsprechenden Transaktion die gewünschte Funktion mit der Maus an, drücken Sie dann (während der Mauscursor auf diese Funktion zeigt) die Taste F1. In einem Popup stellt das SAP System Ihnen nun den internen Funktionscode zur Verfügung. Die Enter-Taste, ohne speziellen Funktionscode, kann durch ein OK am Ende des URL-

	Strings ausgelöst werden.
Aufruf eines InputScript	In Verbindung mit dem InputAssistant kann auch ein InputScript gestartet werden. Hierzu wird der OK:fcode um die Angabe process=... ergänzt. Beispiel: **<A href="SAP://feld1:wert1;feld2:wert2; feld3:wert3;OK:fcode,process=displaymaterial.txt">** Eventuelle using-Parameter für das InputScript können in der SAP-URL davor durch **U[name]:wert** gesetzt werden. Es ist ebenfalls möglich, an interne GuiXT Variablen durch die Notation **V[name]:wert** einen Wert zuzuweisen, der dann im InputScript zur Verfügung steht.
JavaScript	Insbesondere wenn die html-Seite Eingabefelder enthält, ist es sinnvoll, über JavaScript die SAP-URL zu bilden und an SAP zu übergeben (siehe Beispiel3).
Beispiel 1	**<A HREF="SAP://OK:PR03"><IMG SRC="Vorschuss.jpg"></A>** Bei Klick auf das Bild `Vorschuss.jpg` wird die Funktion `PR03` im SAP ausgelöst.
Beispiel 2	**<A HREF="SAP://Material:00012345;Werk:0021;OK">Schraube 8x15mm</A>** Bei Klick auf den Text `"Schraube 8x15mm"` werden die Felder `"Material"`und `"Werk"` gefüllt und die **Enter-Taste** ausgelöst..
Beispiel 3	**Ausschnitt HTML-Seite:** **<form name="form"> Customer <input type="text" size="20"name="Customer" value="&[::Debitor]"> <input type="button" name="Display" value="Display " onclick="DisplayCustomer()"> </form>** **<script language="JavaScript"><!--** **function toGuiXT(string) {window.location.replace("SAP://" + string);}** **function DisplayCustomer() {toGuiXT("U[Customer]:" + document.form.Customer.value + ";OK:/NVD03,process=ShowCustomerData.txt");}** **// --></script>** Bei Klick auf den Button "Display" wird die Transaktion VD03 mit dem InputScript "ShowCustomerData.txt" aufgerufen. Die eingegebene Kundennummer wird dem Script als Parameter **Customer** übergeben. Falls die html-Seite mit dem `View`-Befehl als template aufgerufen wird, erscheint als Vorschlagswert für die Kundennummer der zuletzt im Feld Debitor eingegebene Wert.

Vergleichen Sie hierzu auch das Viewer Tutorial 5 Datenerfassung mit html und Javascript.

Dokumente öffnen

Mit `View` können Sie beliebige Dokumente als separate Applikation öffnen:

```
View "Name der Datei"
```

Hierdurch wird die gleiche Applikation geöffnet, die im Windows Explorer bei Doppelklick des Benutzers auf der Datei gestartet wird.

Beispiele: .doc oder .pdf Dateien. Sie können auch .exe oder .bat Dateien starten.

Zusatzoptionen

`Operation=`	Die auszuführende Operation. Default ist "Open". Welche weiteren Operationen möglich sind, ist pro Dokumenttyp unterschiedlich. Häufig sind "Edit" und "Print" ebenfalls unterstützt.
`Parameters=`	Für .exe und .bat Dateien: Ein String mit Parameterwerten
`Directory=`	Default-Verzeichnis

Tips & Tricks

- Die Bereichsmenüs im SAP können durch ein html-Dokument optisch ansprechend und schnell bedienbar aufbereitet werden
- Zur Anwenderschulung kann man spezielle html Dokumente mit vorgeschlagenen Eingabewerten erstellen.
  Vergleichen Sie hierzu das Viewer Tutorial 7 Hilfstexte und Grafiken einblenden mit ViewHelp.
- In Verbindung mit dem InputAssistant können Sie auch ein html-Dokument bei Drücken einer selbst definierten Drucktaste einblenden, z.B.:

GuiXT Script:

```
Pushbutton (Toolbar) "Dokumentation einblenden"
Process="ActivateView.txt"

if V[GuiXT_Show_View=X]
Set V[GuiXT_Show_View] "N"
View (1,60) (20,100) "C:\GuiXT\html\docu100.html"
endif
```

InputScript `ActivateView.txt`:

```
Set V[GuiXT_Show_View] "X"
```

- Sie können mit dem Viewer ein Bild einblenden, um die Wartezeit während der Ausführung eines längeren Inputscript zu überbrücken:

```
//GuiXT Script
View (1,1) (10,40) "pleasewait.gif" -floating
returnwindow="h_wait"
Screen ...
```

```
        ...
        Enter
        ...

// Last screen


Screen ...
View closewindow="&V[h_wait]"
Enter
```

4.48. Windowposition

Funktion	Mit der `WindowPosition`-Anweisung legen Sie eine neue Position für einen popup Bildschirm fest.
Beispiel	`WindowPosition (2,10)`
Format	`WindowPosition (Zeile,Spalte)` Legt die Position der linken oberen Ecke des popup-Windows fest.

4.49. Windowsize

Funktion	Mit der `Windowsize`-Anweisung legen Sie eine neue Grösse für einen popup Bildschirm fest. Das ist insbesondere dann nützlich, wenn Sie zusätzliche Elemente auf einem zu kleinen popup unterbringen möchten.
Beispiel	`Windowsize (12,60)`
Format	`Windowsize (Zeilen,Spalten)` Gibt dem popup die gewünschte Grösse.

5. Systemvariablen

Variable	Inhalt	Beispiel
Windowsvariablen		
&%[TMP]	Temporäres Windowsverzeichnis	*C:\Temp*
Allgemeine Statusinformationen		
&V[_user]	Benutzername	*SCHMIDT*
&V[_client]	Mandant	*001*
&V[_language]	Sprache	*E*
&V[_transaction]	Transaktionscode	*MM03*
&V[_transactionid]	transaction id: Transaktionscode + Sequenznummer	*MM03.7*
&V[_title]	Titel des SAP-Bildschirms	*Material anzeigen*
&V[_database]	SAP Systemname	*C11*
&V[_ashost]	Name des Applikationsservers	*pwdf0042*
&V[_program]	Programmname	*SAPMF05A*
&V[_dynpro]	Dynpronummer	*0100*
&V[_last_fcode]	Zuletzt benutzter Funktionscode	*/NVA02*
&V[_homedir]	SAP GUI Verzeichnisname	*C:\Program Files\SAP\FrontEnd\sapgui*
&V[_hostname]	Name des Frontend PC	*dw0158*
&V[_sapgui_version]	SAP GUI Version	*640*

&V[_sapgui_patch]	SAP GUI Patch level	*21*
Windowgrösse		
&V[_windowsize_x] &V[_windowsize_y]	Fensterbreite in Pixel Fensterhöhe in Pixel	*1250* *850*
&V[_screensize_x] &V[_screensize_y]	Bildschirmauflösung (Breite) Bildschirmauflösung (Höhe)	*1920* *1200*
Cursorposition		
&V[_cursorrow] &V[_cursorcol]	Cursorposition: Zeile und Spalte Verfügbar in einem InputScript, wenn der Benutzer den Cursor in ein Eingabefeld gestellt hat	*12* *30*
&V[_cursorfieldname] &V[_cursorfieldtechnname]	Cursorposition: Feldname Technischer Feldname	*Buchungskreis* *BKPF-BUKRS*
&V[_cursortabname] &V[_cursortabtechname]	Cursorposition: Tabellenname Technischer Tabellenname	*Alle Positionen* *SAPMV45A_TCTRL_U_ERF_AUFTRAG*
&V[_cursorcolname] &V[_cursorcoltechname]	Cursorposition: Spaltenname Technischer Spaltenname	*Material* *RV45A-MABNR*
&V[_tabrow] &V[_tabcol]	Cursorposition: Nummer der Tabellenzeile und der Tabellenspalte, jeweils bei 1 beginnend Verfügbar in einem InputScript, wenn der Benutzer den Cursor in eine Tabellenzelle gestellt hat	*5* *6*
&V[_listcursorrow]	Cursorposition in Listen: Zeile (bei 0 beginnend)	*7*
&V[_listcursorcol]	Cursorposition in Listen: Spalte (bei 0 beginnend)	*48*
Nachrichten		
&V[_lastwarning]	Text der letzten Warnungsmeldung während der Verarbeitung eines	*W: Datum liegt in der Zukunft*

	InputScripts	
&V[_lasterror]	Text der letzten Fehlermeldung während der Verarbeitung eines InputScripts, in Klammern ergänzt um Transaktion, Programm und Dynpronummer	*E: Buchungsschlüssel X2 ist nicht vorgesehen (FB01,SAPMF05A,0300)*
&V[_message]	Nachrichtentext (Information)	*Beleg 100000444 wurde im Buchungskreis 1000 gebucht*
&V[_msgid]	Nachrichten Id	*05*
&V[_msgno]	Nachrichtennummer	*055*
&V[_msgtype]	Nachrichtentyp	*W*
&V[_exception]	Text der "exception" nach einem Funktionsaufruf mit Option **-try**	*Material 15088 nicht gefunden*
Inhalte bei Listen		
&V[_listline(0)] &V[_listline(1)] &V[_listline(2)] ...	Inhalte der Zeilen in einer Liste Hinweis: Senkrechte und waagerechte Linien in der Liste sind intern durch Ziffern (4,5,..) dargestellt	*5M-06 10000 05*
&V[_listline]	Inhalt der Listenzeile, in der sich der Cursor befindet	*5M-02 28000 09*
&V[_liststring]	String innerhalb der aktuellen Listenzeile, auf den der Cursor positioniert ist	*28000*
Blätterinformationen bei Listen und Tabellen		
&V[_listfirstvisiblerow]	Erste auf dem aktuellen Bildschirm angezeigte Zeile bei Listen oder Tabellen	*252*

&V[_listlastvisiblerow]	Letzte auf dem aktuellen Bildschirm angezeigte Zeile bei Listen oder Tabellen	*280*
&V[_listlastrow]	Gesamtzahl der Zeilen einer Liste oder Tabelle	*578*
Datum und Uhrzeit		
&V[today_ymd]	Tagesdatum in der Form: Jahr Monat Tag	*20021224*
&V[today_hms]	Uhrzeit in der Form: Stunde Minute Sekunde	*154810*
&V[today...]	Aufbereitungsvarianten sowie Rechnungen mit today+... oder today-... sind möglich. In den Rechnungen können + oder - sowie d=Tage, h=Stunden, m=Minuten, s=Sekunden verwendet werden. Spezielle Userabhängige Aufbereitung [today_user]	&V[today_m/d/y] --> 12/24/2002 &V[today_h:m] --> 15:48 &V[today_d.m.y h:m] --> 24.12.2002 15:48 &V[today+1_m/d/y] --> 12/25/2002 &V[today+14_m/d/y] --> 1/7/2003 &V[today-1_m/d/y] --> 12/23/2002 &V[today+6h_h:m] --> 21:48 &V[today+12h_m/d/y] --> 12/25/2002
&V[today_user]	Tagesdatum in der Form, wie sie der Benutzer eingestellt hat. Bitte beachten: Hierzu benötigen Sie die Angabe eines gültigen RFC-Benutzers im GuiXT Profile mit entsprechenden Berechtigungen zum Ausführen des Funktionsbausteins BAPI_USER_GET_DETAIL	*25.12.2010* *12/25/2010*
Variablen in den Profile-Verzeichnisnamen		
&user	Benutzername	*SCHMIDT*

&client	Mandant	*001*
&database	SAP System Id	*C11*
&winuser	Windows Benutzername	*Smith*

Weitere Hinweise zum Thema Variablen: Statusabfragen mit if/else/endif und Special Topic Variablen, Berechnungen, Vergleiche

© by Synactive GmbH

6. Input Assistant

6.1. Was leistet der InputAssistant ?

Der InputAssistant unterstützt folgende Funktionen:

- Eingabehistorie
- Lokale Wertehilfe
- InputSet's
- Einbildtransaktionen
- Verarbeitung ab einem File

6.2. Eingabehistorie

Eingabehistorie

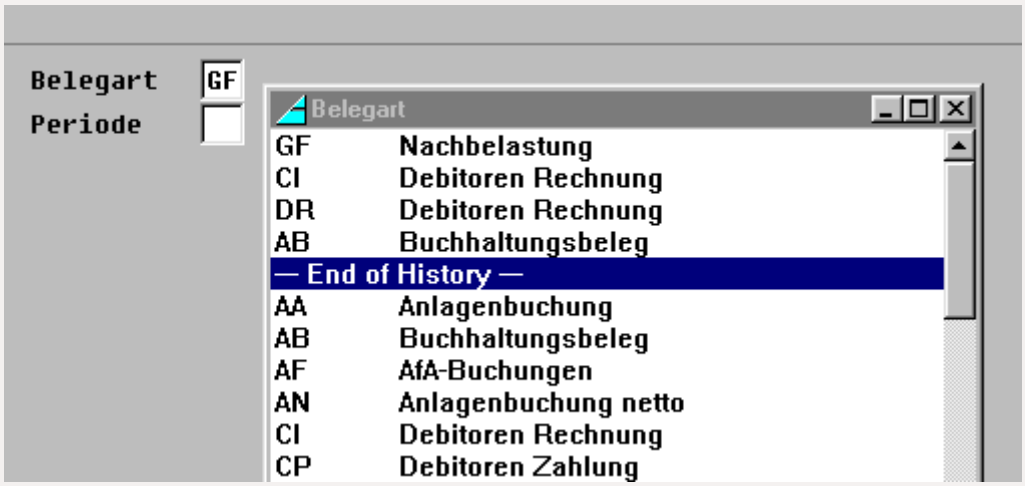
GuiXT InputAssistant speichert auf Wunsch eine Histore Ihrer Eingabewerte ab. Durch Drücken der rechten Maustaste oder der Anwendungstaste können Sie pro Eingabefeld Ihre bisherigen Eingabewerte anzeigen lassen. Mit den Pfeiltasten und Blättern können Sie den gewünschten Wert suchen und durch Enter oder Doppelklick in das Eingabefeld übernehmen.

Die Auswahlliste verschwindet wieder, wenn Sie

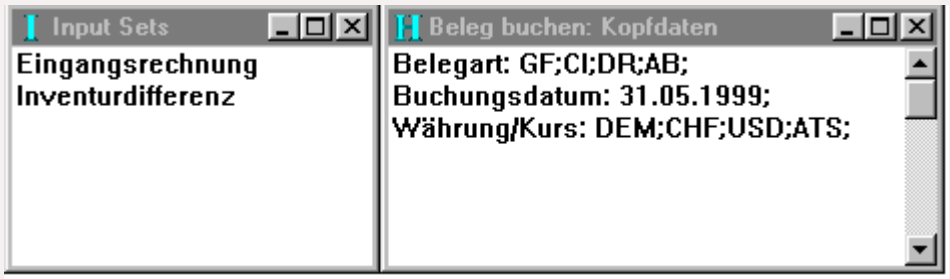
- nochmal die Anwendungstaste drücken, oder
- mit der Maus ein anderes Feld anklicken, oder
- einen Wert aus der Auswahlliste übernehmen

Die Sortierung der Eingabewerte erfolgt nach der zeitlichen Reihenfolge, d.h. der zuletzt eingegebene Wert wird in der ersten Zeile angezeigt:

Falls Sie die lokale Wertehilfe (s.u.) installiert haben, werden dabei auch die Langtexte zu den jeweiligen Werten in die Eingabehistorie eingeblendet:



Die Eingabehistorie für alle Felder eines SAP Bilds sehen Sie komprimiert auch in der **History** Anzeige von GuiXT:



Auch dort können Sie über einen Mausklick einzelne Werte abrufen.

Wo sind die Eingabewerte abgelegt?	Die Eingabehistorie wird intern pro SAP Bild in einer Datei mit der Endung **.inp** abgelegt. Das Verzeichnis für die Ablage der .inp-Dateien können Sie in guixt.ini durch den Parameter **History** vorgeben. Es ist oft sinnvoll, dabei den Systemnamen des SAP Systems als Variable (&database) mitzugeben, um die Eingabewerte pro System zu trennen: **History C:\GuiXT\History\&database** In dem System TST z.B. werden die .inp-Datein dann im Verzeichnis C:\GuiXT\History\TST geführt.
Aktivieren und Deaktivieren der Abspeicherung	In GuiXT können Sie durch **Options->History Update active** die Abspeicherung der Eingabewerte dynamisch ein- und ausschalten. Sie können es auch statisch als Parameter in guixt.ini festlegen: **HistoryUpdate Yes/No**
Manuelle Pflege der Eingabehistore	Gehen Sie im SAP System auf das Bild, dessen Eingabehistorie Sie verändern möchten. Klicken Sie dann im GuiXT Menü auf **Edit->History**. Es wird der Texteditor (Notepad) für die History-Datei gestartet, und Sie können z.B. einzelne Werte löschen. Achten Sie dabei bitte auf die Einhaltung des Formats: **Feldbezeichnung: Wert1;Wert2;Wert3;Wert4;**

	z.B. **Buchungskreis: 001;008;003;**

6.3. Lokale Wertehilfe

Lokale Wertehilfe	Die zentrale Wertehilfe des SAP-Systems können Sie durch den GuiXT Input Assistant um eine lokale Variante ergänzen, die in der praktischen Arbeit mit SAP folgende Vorteile bietet: • **Sie wird sehr schnell, ohne Inanspruchnahme des zentralen SAP-Systems, angezeigt.** Das gestattet einerseits dem Benutzer, die Wertehilfe als schnelle Eingabemöglichkeit zu benutzen, was bei der zentralen Wertehilfe durch die längere und kaum vorhersehbare Antwortzeit (z.B. 1 oder 3 Sekunden für die Anzeige und für jedes Blättern) schlecht möglich ist. • **Das SAP System wird durch die lokale Wertehilfe entlastet** • **Die lokale Wertehilfe kann ohne Änderung der SAP Datenbank angepaßt und auf einzelne Benutzergruppen zugeschnitten werden** Zum Beispiel können Sie Werte entfernen, die nicht oder nicht mehr benutzt werden, oder Texte anpassen. Die hierzu benötigten Dateien können Sie aus dem SAP-System generieren (siehe weiter unten) und entweder auf einzelnen PCs oder auf einem Server im lokalen Netz ablegen. Der Speicherplatz (pro Sprache) beträgt insgesamt etwa 5-10 Megabyte. Das Verzeichnis für die Dateien der lokalen Wertehilfe können Sie in guixt.ini oder GuiXT Profile durch den Parameter **Domains** festlegen, z.B. **Domains \\P80590\GuiXT\Domains** Default hierfür, falls Sie keine **Domains** Angabe in guixt.ini haben, ist das Unterverzeichnis Domains in dem Pfad, aus dem GuiXT gestartet wurde, also z.B. C:\GuiXT\Domains.
Funktion	Die lokale Wertehilfe benötigt folgende Dateien: • Pro Domäne xxxxx, für die eine lokale Wertehilfe verfügbar sein soll, eine Wertehilfedatei mit dem Namen **dom.D.xxxxx.txt** Also zum Beispiel **dom.D.BUKRS.txt** Buchungskreise **dom.D.WAERS.txt** Währungen **dom.D.GSBER.txt** Geschäftsbereiche Für englischsprachige Wertehilfen entsprechend **dom.E.BUKRS.txt**,

dom.E.WAERS.txt.

- Zwei Domänendateien für die Zuordnung der SAP Felder zu Domänen:
  domText.D.txt
  domTextScreen.D.txt

Sämtliche Dateien können maschinell aus dem SAP System erzeugt werden (ABAP Programme **ZGUIXT12** und **ZGUIXT16**). Optional können die Dateien auch manuell gepflegt werden, um die zentrale Wertehilfe an Ihre Bedürfnisse anzupassen (zum Beispiel nur eine Teilmenge der Werte).

Sie können auch für Domänen, zu denen sehr viele Werte existieren und zu denen SAP bei F4 statt einer festen Wertemenge eine Matchcode-Suchhilfe anbietet (bei der Materialnummer z.B.), die gebräuchlichsten Werte in Form einer Wertehilfedatei manuell oder maschinell zusammenstellen.

Generieren der Wertehilfedateien

Sie starten das ABAP Programm **ZGUIXT16** im SAP System:

Wählen Sie eine einzelne Domäne oder eine Menge von Domänen aus. Über den Parameter **Maximum number of values** selektieren Sie nur diejenigen Wertehilfen, die aus einer kleineren als der angegebenen Anzahl von Werten bestehen.

Die Laufzeit für eine einzelne Domäne ist sehr gering (Größenordnung 1 Sekunde). Die erzeugte Datei ist im allgemeinen ca 1-10k groß.

Eine Gesamtgenerierung aller Wertehilfedateien benötigt etwas mehr Zeit (Größenordnung 30 Minuten). Der gesamte Platzbedarf für alle Dateien (bei Parameter 400) liegt in einem IDES-System unter 10 Megabyte.

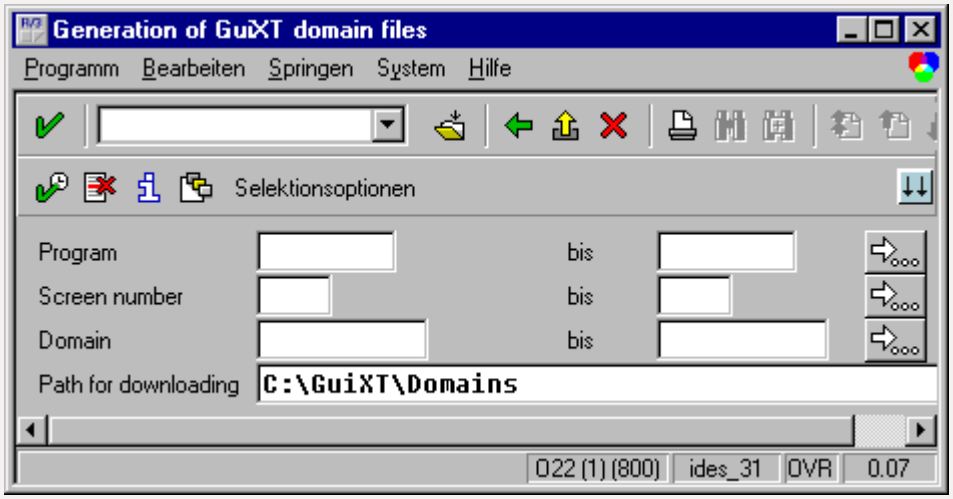
Das Programm **ZGUIXT16** generiert die Wertemengen nur für diejenigen Domänen, für die im Data Dictionary (**SE11**) eine Wertemengentabelle spezifisiert ist.

Das Programm **ZGUIXT16** können Sie sich von unserer Download-Seite herunterladen.

Generieren der

Sie starten das ABAP Programm **ZGUIXT12** im SAP System:

Domänendateien



Eine Selektion nach Programm, Dynpronummer oder Domäne ist nur zu Testzwecken vorgesehen. Zur produktiven Nutzung sollte ein Gesamtlauf ohne Einschränkungen erfolgen.

Die beiden erzeugten Dateien sind in einem IDES-System zusammen etwa 0.5 Megabyte groß. Die Laufzeit über alle Dynprosr des SAP-Systems liegt in der Größenordnung von 30 Minuten.

Das Programm **ZGUIXT12** können Sie sich von unserer Download-Seite herunterladen.

Struktur der Wertehilfedateien

Die Wertehilfedateien sind analog zur F4-Wertehilfe von SAP aufgebaut. Pro Wert enthält die Datei eine Zeile. Diese beginnt mit dem jeweiligen Wert, anschließend folgen die übrigen Spalten (meist der zugehörige Text). Die Spalten sind durch zwei Schrägstriche // voneinander getrennt. Beispiele (Domänen **WAERS** und **BUKRS**):

ADP //Andorianische Peseta
AED //Vereinigte Arabische Emirate Dirham
AFA //Afghani
ALL //Albanische Lek
AMD //Armenische Dram
ANG //Antilianische Gulden
AOK //Angolesische Kwanza
ARA //Argentinische Austral
ATS //Österreichische Schilling
AUD //Australien Dollar
AWG //Arubanische Gulden
AZM //Aserbaidschanische Manat
...

0001 //Buchungskreis 0001 //Walldorf //DEM
1000 //IDES AG //Frankfurt //DEM
2000 //IDES UK //London //GBP
2100 //IDES Portugal //Lisbon //PTE
2200 //IDES France //Paris //FRF

	2300 //IDES España //Barcelona //ESP **3000** //IDES US INC //New York //USD **4000** //IDES Canada //Toronto //CAD **4100** //SAP Australia //Melbourne //AUD **4200** //IDES Australia //Melbourne //AUD **5000** //IDES Japan //Tokyo //JPY **6000** //IDES Mexico //Mexico DF //MXN ...
Struktur der Domänendateien	Die Datei **domText.D.txt** enthält pro Feldbezeichnung die am häufigsten verwendete Domäne. Format einer Zeile: **Text: Domäne** Beispiele: **Finanzamt:FZAMA** **Finanzamtsnummer:FZAMN** **Material:MATNR** **Materialart:MTART** In der Datei **domTextScreen.D.txt** sind alle Ausnahmen zu finden, d.h. alle Felder mit einer anderen Domäne, als sich aus ihrer Bezeichnung gemäß Datei **domText.D.txt** ergeben würde. Dem Feldtext sind dabei Programmname und Dynpronummer vorangestellt, z.B.: **MP001200.2000.Finanzamt:FZAMN**
Wann ist eine Neugenerierung der Dateien nötig?	Einzelne Wertehilfedateien müssen gegebenenfalls nach Customizing-Änderungen neu generiert werden, z.B. nachdem eine neue Materialklasse oder ein neuer Geschäftsbereich aufgenommen wurde. Eine Neugenerierung der Domänendateien ist nach Modifikationen an Dynpros oder nach Releasewechsel nötig, da sich eine der Feldbezeichnungen wie Material, Geschäftsbereich, Buchungsschlüssel, ... geändert haben könnte oder neue Domänen dazugekommen sind.
Pufferung der lokalen Dateien	Die Domänendatei **domText.D.txt** und **domTextScreen.D.txt** werden von GuiXT nach dem ersten Einlesen im Speicher gehalten. Die Wertehilfedateien dagegen werden jeweils bei Bedarf hinzugelesen. Änderungen von **domText.D.txt** oder **domTextScreen.D.txt** werden daher erst wirksam, nachdem Sie sich von SAP abgemeldet und dann GuiXT beendet und neu gestartet haben. Alle übrigen Dateien (Wertehilfedateien) werden jeweils neu gelesen. Falls die Dateien auf einem zentralen Fileserver im Netz abgelegt sind, können Sie im GuiXT die

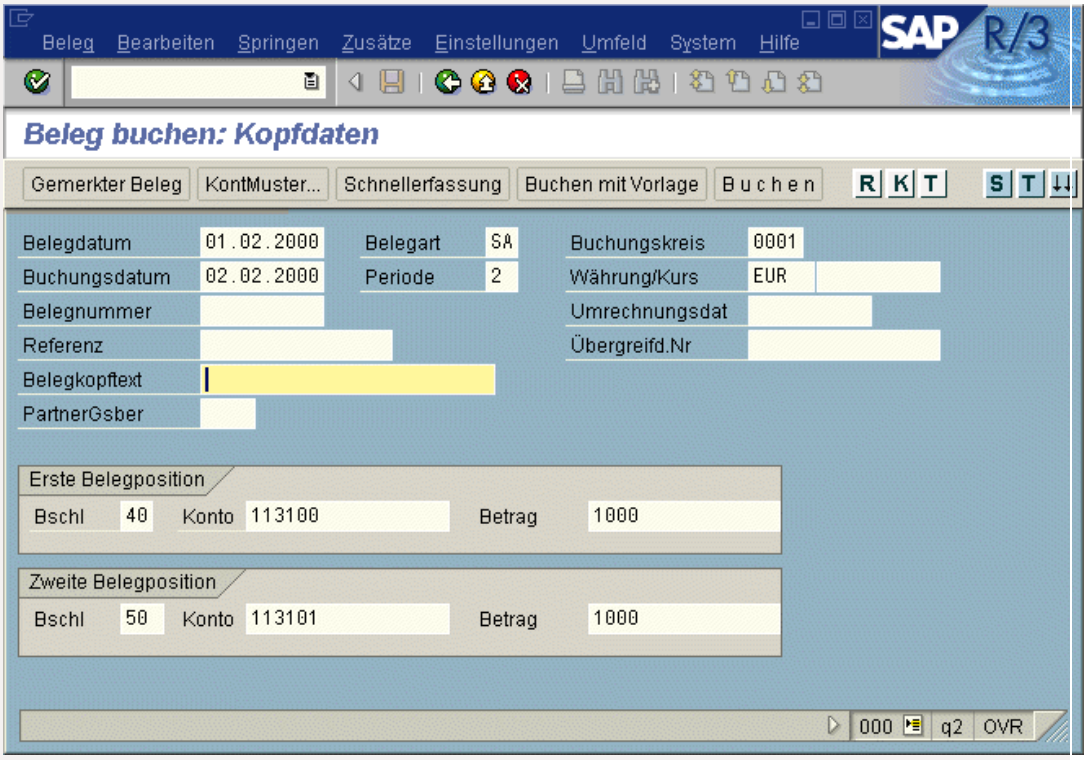
	Cache-Funktion einschalten und die Dateien dadurch lokal replizieren (sie Dokumentation zum GuiXT Profile).
Anlegen spezieller eigener Werthilfedateien	Mit wenig Aufwand können Sie für einzelne Felder eine eigene spezielle Wertehilfe erfassen; auch dort, wo im SAP System keine Standard-F4-Hilfe vorgesehen ist. So gehen Sie vor: • Sie denken sich einen eigenen Domänennamen für das betreffende Feld aus und tragen diese in die Datei **domTextScreen.D.txt** ein, z.B. **SAPMF05A.0100.Bschl:GUIXT_BSCHL** • Unter diesem Domänennamen erfassen Sie eine Datei **dom.D.GUIXT_BSCHL.txt** in dem oben beschriebenen Format Hierbei stehen einige Sonderoptionen zur Verfügung: • **Historysize=** Durch eine Zeile mit der Angabe **Historysize=n**, wobei n eine Zahl 0,1,2... ist, beschränken Sie die Anzeige der Eingabehistorie. Insbesondere unterdrückt **Historysize=0** die Anzeige der Eingabehistorie vollständig. Das ist dann sinnvoll, wenn Sie eine sehr kleine Wertemenge (z.B. 5 Werte) zur Eingabe anbieten. • **WindowHeight=, WindowWidth=, WindowXpos=, WindowYpos=** Legt die Grösse und Position des anzuzeigenden Windows fest (alle Angaben in Pixel) • **//Input=** Sie können auch die automatische Eingabe für mehrere Felder gleichzeitig auslösen, indem Sie folgendes Format verwenden (Beispiel): **Beschreibender Text 1 //Input=Bschl:40;Konto:113100** **Beschreibender Text 2 //Input=Bschl:31;Konto:40080002** In diesem Fall werden die beschreibenden Texte in der Listbox angezeigt. Bei Auswahl einer Zeile werden die angegebenen Felder mit den betreffenden Werten gefüllt. Alle bei der Definition eines **Input Set** gültigen Optionen sind möglich, insbesondere auch Positionierung des Cursors und Eingabe in einzelne Zellen einer Tabelle. Sie können beispielsweise für die erste Spalte einer Tabelle eine eigene Wertehilfe definieren und bei Auswahl durch den Benutzer einige weitere Spalten der laufenden Zeile füllen (als Zeilennummer dabei * angeben).
Checkliste	Bei Problemen hilft vielleicht die Checkliste weiter

6.4. InputSet's direkt im R/3 Bild

Funktion

Input Sets bieten die Möglichkeit, vordefinierte Eingabewerte abzulegen und über einen Mausklick abzurufen. Pro Bildschirmmaske können Sie mehrere Input Sets definieren.

Die Input Sets können sowohl individuell pro Benutzer als auch für Benutzergruppen definiert werden. Sie werden in der Drucktastenleiste ganz rechts angezeigt: in Blau die individuell definierten Input Sets mit den jeweiligen Kennbuchstaben, z.B. K R, in Gelb die allgemein zur Verfügung stehenden, z.B. R K.



Ein Klick auf die rechte Maustaste zeigt Ihnen den Namen des Input Set sowie die Feldnamen und Inhalte:

Beispiel

Persönliche Input Sets definieren Sie, indem Sie im GuiXT Window auf **Edit->History** klicken und dort die Werte erfassen. Jedes Input Set beginnt mit einem Titel in eckigen Klammern, anschließend pro Eingabewert eine Zeile mit der Feldbezeichnung, Doppelpunkt und dem einzugebenden Wert:

```
[Rechnungseingang]
Buchungskreis: 0001
Belegart:RE
Währung/Kurs: DEM
```

	Der erste Buchstabe des Titels, z.B. **R**, wird dann von GuiXT in der Drucktastenleiste angezeigt. Der Dateinamen für allgemein benutzbare Input Sets ist der gleiche. Die Datei kopieren Sie in das im GuiXT Profile angegebene Verzeichnis "Shared Input Sets".

Feldbezeichnungen	Folgende Feldbezeichnungen sind möglich
	
	• **Feldtexte**
	`Buchungskreis: 0001`
	`Belegart: KN`
	
	• **Direkte Positionsangabe**
	Im Format #[Zeile,Spalte,Feldlänge]:
	`#[5,20,18]:00058102`
	
	• **Technische Feldnamen (sprachunabhängig)**
	`BKPF-BLART: SA`
	
	• **Tabellenfelder**
	Im Format cell[Tabellenname,Spaltenname,Zeile]. Bei Tabellen ohne Titel bitte "table" als Name verwenden.
	`cell[Alle Positionen,ME,1]: ST`

Auslösen der Enter-Taste	Im Anschluss an die Eingabe der Werte können Sie durch `OK` automatisch die Enter-Taste drücken, z.B.:
	`Werk: 0012`
	`Lagerort: 0003`
	`OK`
	
	Sie können auch einen Funktionscode (Transaktionscode, internen Code oder F-Taste) mitgeben, z.B.
	`Werk: 0012`
	`Lagerort: 0004`
	`OK: /9`
	
	Welche Codes hier möglich sind, finden Sie in der Dokumentation zur `Pushbutton`-Anweisung.
	
	Es ist auch möglich, nach Auslösen der Enter-Taste die nachfolgende Maske ebenfalls mit Werten zu versehen. Im Unterschied zu einem Input Script (`Pushbutton`-Anweisung, `process=`) werden die Masken in diesem Fall sichtbar prozessiert, sodass längere Ketten hier nicht unbedingt sinnvoll sind.

Positionierung des Eingabecursors	Durch die Eingabe in ein Feld wird auch der Eingabecursor dorthin gesetzt. Nach dem Prozessieren eines Input Set steht daher der Cursor in dem zuletzt im InputSet aufgeführten Feld. Sie können den Cursor explizit positionieren, ohne eine Eingabe auszulösen. Hierzu nennen Sie den Feldnamen und dann als Wert "`@Cursor`", z.B.: `Kennwort: @Cursor`
Aufruf der lokalen	Durch den Wert "`@History`" können Sie für ein Feld die Eingabehistorie und

Wertehilfe

lokale Wertehilfe anzeigen. Das ist insbesondere dann nützlich, wenn Sie zu einem Eingabefeld (Standardfeld oder eigenes InputField) durch eine "Image"-Anweisung mit "Input=..." die lokale Wertehilfe als kleine Ikone nach dem Feld anbieten wollen:

```
Image (..,..) "valuehelp.gif" Input="Abteilung:@History"
```

Cursor-gesteuerte Eingabe in Tabellen

Bei der Eingabe in Tabellen ist es meist sinnvoll, die Tabellenzeile durch den Benutzer vorgeben zu lassen. Je nach Positionierung des Eingabecursors wird die Eingabe dann in der ausgewählten Zeile vorgenommen. Falls der Benutzer den Cursor nicht innerhalb der Tabelle positioniert hat, erfolgen die Einträge in die erste Tabellenzeile.

Geben Sie in der cell-Notation hierzu als Zelle * an, z.B.

```
cell[Alle Positionen,Material,*]: 10007684
cell[Alle Positionen,Auftragsmenge,*]: 1
cell[Alle Positionen,ME,*]: ST
```

Bei mehrzeiligen Einträgen ist auch die Angabe *+1 als Zeile möglich. Aber bitte beachten Sie hierbei, daß durch die Eingabe des Wertes der Cursor dann um eine Zeile nach unten positioniert wird, sodaß die Angabe *+1 nur bei dem jeweils ersten Feld pro neuer Zeile erfolgen darf:

```
cell[Alle Positionen,Material,*]: 10007684
cell[Alle Positionen,Auftragsmenge,*]: 1

cell[Alle Positionen,Material,*+1]: 10007685
cell[Alle Positionen,Auftragsmenge,*]: 10

cell[Alle Positionen,Material,*+1]: 10007686
cell[Alle Positionen,Auftragsmenge,*]: 5
```

Zum Schluß können Sie den Eingabecursor in die nächstfolgende Zeile stellen:

```
cell[Alle Positionen,Material,*+1]: @Cursor
```

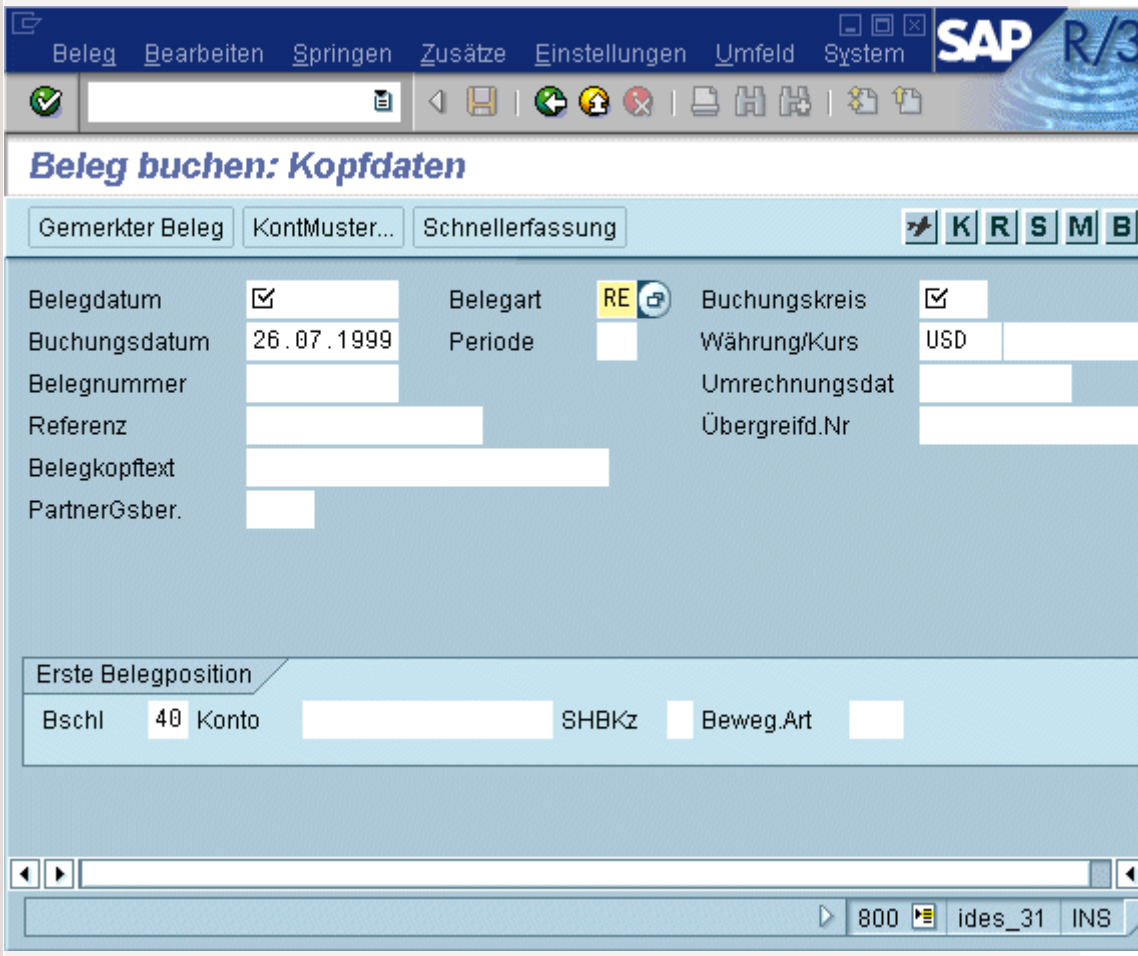
Falls es sich um eine tabellenartige Darstellung handelt, ohne daß ein Tablecontrol verwendet wäre (insbesondere in SAP Rel. 3.1), so verwenden Sie ebenfalls den Index *, z.B. Material(*). Dadurch erfolgt die Eingabe in der Zeile, in der der Eingabecursor gerade steht. Auch hier ist die Notation *+1 ebenfalls unterstützt.

Eigene Bilder statt Anfangsbuchstaben

Sie können in der Definition eines Input Set auch eine eigene kleine Bilddatei angeben, die dann statt des Buchstabens angezeigt wird, z.B. :

```
[Reisebeleg] Image=C:\GuiXT\Images\travel.gif
Belegart: T5
Buchungskreis: 0001
```

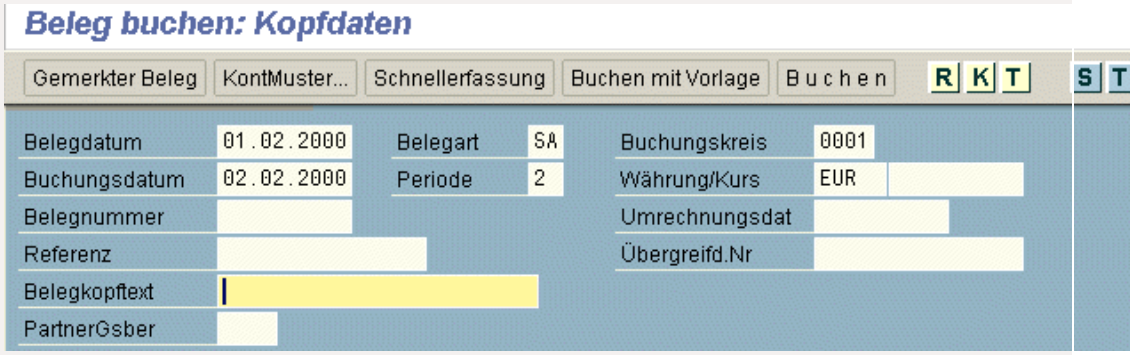
Die Größe sollte 18x18 Pixel betragen. Die Umrandung (3D-Effekt) muß, falls gewünscht, im Bild enthalten sein.



Gemeinsam genutzte InputSets für mehrere Benutzer

Im GuiXT-Profile können Sie ein zusätzliches Verzeichnis angeben: "Shared input sets". Sie können hier Input Sets definieren, die von allen Anwendern oder von Anwendergruppen eingesetzt werden sollen. Diese Input Sets werden durch Ikonen einer anderen Farbe dargestellt (gelb anstelle von blau).

Shared Input Sets



Gemeinsame genutzte Input Sets können pro Bild **sprachabhängig** oder **sprachunabhängig** gepflegt werden. GuiXT sucht jeweils zunächst nach eine .inp-Datei, die den Sprachenschlüssel im Dateinamen enthält (Aufbau identisch wie bei den Skriptnamen). Wird diese nicht gefunden, wird eine sprachunabhängige Datei gesucht (Dateinamen durch Weglassen des Sprachenschlüssels gebildet).

Image und View Anweisungen	Eingabewerte können auch bei einem Image oder in einem html-Dokument hinterlegt werden, siehe die GuiXT Script-Anweisungen **Image** und **View**.
Externe Anwendungen	Eingabewerte können auch von externen Anwendungen über GuiXT an die SAP Session gesendet werden. Hiezu bietet GuiXT ab Version 2003 Q2 3 ein einfaches Kommandozeilen-Interface. Die Eingaben werden jeweils in der aktuellen SAP Session vorgenommen. Ist keine Session aktiv, geschieht nichts. Bei Bedarf kann die externe Anwendung zunächst selbst über einen SAP Shortcut (.sap Datei) eine Session starten. Die Übermittlung der Daten erfolgt über den (manuellen oder programminternen) Aufruf guixt input="inputstring" Dabei ist inputstring so aufgebaut wie für den Parameter input= der **Image** Anweisung beschrieben. Es ist möglich, Eingabewerte zu setzen, Aktionen durchzuführen und InputScripts zu prozessieren. Beispiel 1: guixt input="Debitor:1000;OK" Hierdurch wird der Wert "1000" in das Feld "Debitor" gesetzt und "Enter" gedrückt. Beispiel 2: guixt input="U[CUST]:1000;OK:/NVD03,process=vd03.txt" Transaktion VD03 wird aufgerufen und das InputScript "vd03.txt" aktiviert. Als Parameter "CUST" wird dem InputScript der Wert "1000" mitgegeben. Beispiel 3: Sie legen eine Batchdatei "showcustomer.bat" an; Inhalt ist guixt input="U[CUST]:%1;OK:/NVD03,process=vd03.txt" Jetzt kann durch Aufruf von showcustomer 1000 die in Beispiel 2 beschriebene Aktion ausgelöst werden (Anzeige Kundenstammsatz

6.5. InputScripts

InputScript	Ein InputScript ist eine Textdatei mit beliebigem Namen, z.B. C:\GuiXT\Scripts\CreateMaterialSimple.txt. Sie besteht im einfachsten Fall aus der Angabe der prozessierten Masken (**Screen**-Anweisung), dem Setzen von Werten (**Set**) und der

Navigation innerhalb der Transaktion (Enter). Falls kein Pfadname bei process= spezifiziert ist, wird das InputScript im aktuellen GuiXT Script Verzeichnis gesucht.

Beispiel:

```
Screen SAPMF05A.0300
  Set F[Betrag]   "&V[Betrag1]"
  Set F[Bschl]    "50"
  Set F[Konto]    "&V[Konto2]"
  Enter

Screen SAPMF05A.0300
  Set F[Betrag]   "&V[Betrag2]"
  Enter "BU"
```

Während der InputAssistant die Transaktion intern durchführt, sucht er im InputScript jeweils die nächste passende Screen-Anweisung und führt die entsprechenden Anweisungen aus. Falls er keine passende Screen-Anweisung findet, wird dem Benutzer das aktuelle Bild angezeigt, und er kann manuell weiterarbeiten. Ist für ein Bild kein Enter angegeben, so wird ebenfalls die Verarbeitung des InputScript beendet.

Bei Fehlermeldungen verzweigt der InputAssistant auf das ursprüngliche Bild, die Fehlermeldung wird dort eingeblendet und der Benutzer kann alle Werte korrigieren.

Warnungen während des Prozessieren eines InputScript werden ignoriert.

Es ist empfehlenswert, in einem InputScript alle benutzten Variablen (d.h. Werte aus Eingabefeldern) als Parameter zu definieren und aus dem GuiXT-Script heraus durch using zu übergeben. Sie gewinnen dadurch mehr Flexibilität bei der Wiederverwendung des InputScript und erhalten weitere Möglichkeiten zur Abfrage der Parameter und zur Steuerung der Transaktion.

Syntax:

```
Parameter PNAME
```

oder

```
Parameter PNAME   "defaultvalue"
```

z.B.:

```
Parameter SPR   "D"
```

Der Defaultwert wird dann eingesetzt, wenn der Parameter in dem GuiXT Script nicht bewertet wurde.

Abfrage von Parametern. Der durch using übergebene Wert des Parameters bzw. der Defaultwert kann durch

```
if U[PNAME=wert]
...
endif
```

abgefragt werden. Beispiel:

```
if U[SPR=D] and U[STDNAME=X]
  Set F[Bezeichnung] "Wartungsplan"
```

```
  Set V[Langtext] " "
endif
```

Durch

```
U[PNAME]
```

allein (ohne Vergleichswert) kann abgefragt werden, ob der Parameter einen Wert ungleich Space besitzt.

Eine eigene Prüfung der Parameter ist in gewissem Umfang ebenfalls möglich. Durch `Return` können Sie ein InputScript beenden und auf das Grundbild zurückgehen. Falls Sie nach `Return` einen Text angeben, wird dieser angezeigt in einem Window mit dem Titel "Information". Den Titel können Sie als zweiten Text bei `Return` setzen. Beispiel:

```
if not U[BUK]
   Return "Bitte Buchungskreis eingeben"
endif

if U[BUK=0001] and not U[WAERS=DEM]
   Return "Nur Währung DEM in Buchungskreis 0001 unterstützt"  "Bitte Währung/Buchungskreis korrigieren"
endif
```

Tips&Tricks

- Zum Testen können Sie über **`Options->Visible Auto Processing`** alle InputScripts auch sichtbar abspielen
- Zum Erstellen von InputScripts können Sie den InputScript Recorder über **`Options->Record InputScript`** und **`Edit->Recorded InputScript`** einsetzen
- Das Setzen von Radiobuttons oder Checkboxes erfolgt über Zuweisung der Werte `"X"` oder `" "`
- Beim Setzen eines Radiobutton ist es nicht nötig (aber möglich), den übrigen Radiobuttons der gleichen Gruppe den Wert `" "` zuzuweisen.

6.6. Einbildtransaktion

Zielsetzung

SAP Transaktionen sind meist Mehrbildtransaktionen, d.h. die Daten werden in mehreren Masken nacheinander eingegeben. Zunächst ein Einstiegsbild, in dem einige Schlüsselfelder einzugeben sind, dann ein oder mehrere oft umfangreiche Masken, in denen der jeweilige Geschäftsvorfall erfasst wird.

Falls der Benutzer pro Bild nur wenige Daten einzugeben hat, empfindet er das als umständlich und langsam, denn er muß

- die tatsächlich einzugebenden Felder in jeder Maske suchen
- nach Datenfreigabe warten, bis die nächste Maske erscheint
- seine Arbeitsweise einer festgelegten Bildfolge und schwankenden Antwortzeiten anpassen

Mit GuiXT+InputAssistant können Sie eine SAP Mehrbildtransaktion mit relativ geringem Aufwand, ohne Risiko und ohne Systemmodifikation, in eine

	Einbildtransaktion verwandeln. Der Benutzer bleibt dann in einer einzigen Maske und kann alle für den Geschäftsvorfall nötigen Daten unmittelbar dort eingeben. Intern wird die ursprüngliche Transaktion mit ihren Masken prozessiert. Die Anwendungslogik und alle Prüfungen bleiben unberührt.
Was müssen Sie tun?	• Mit GuiXT nehmen Sie weitere Eingabefelder auf dem Einstiegsbild der Transaktion oder in einem Menübild auf. Sie fügen eine neue Drucktaste ein, durch die das Prozessieren der gesamten Transaktion ausgelöst wird • In einem speziellen "InputScript" legen Sie fest, wie die Eingaben auf den nachfolgenden (nicht angezeigten) Masken und die Navigation innerhalb der Transaktion aussehen. Der InputAssistant prozessiert dann intern diese Masken und mischt die Eingabewerte ein. Auch bei Fehlermeldungen bleibt der Benutzer auf der von Ihnen gestalteten Maske und kann dort seine Eingaben korrigieren.
Beispiele	• Buchen zweier Buchungszeilen mit FB01 • Anlegen Materialstammsatz MM01 • Anzeigen Materialstammsatz MM03
Allgemeines	Einbildtransaktionen können entweder auf dem Einstiegsbild einer Transaktion oder in einem Menübild definiert werden. Es kann sich übrigens auch um das Einstiegsbild einer anderen Transaktion handeln. Ebenfalls möglich ist, nur den Einstieg in eine komplexe Transaktion(z.B. die ersten beiden Masken) automatisch ablaufen und den Benutzer die restlichen Masken manuell durchführen zu lassen. In beschränktem Rahmen sind eigene zusätzliche Verprobungen der Eingabewerte sowie das Setzen weiterer Felder in Abhängigkeit von den Eingabewerten möglich.

Eingabefelder	Unterstützt sind Felder zur Eingabe von Werten, Ankreuzfelder und Auswahlknöpfe:
	
	`InputField (row1,col1) "Text" (row2,col2) Size="..." Name="..." Default="..."`
	`Checkbox (row,col) "Text" Name="..." Default="..."`
	`Radiobutton (row,col) "text" Name="group.name" Default="..."`
	
	Beispiele:
	
	`InputField (10,40) "Bezeichnung" (10,48) Size="3" Name="Bz"`
	`Checkbox (11,40) "Nur Grunddaten" Name="NurGD" Default="X"`
	`Radiobutton (14,20) "Stück" Name="einheit.st" Default="X"`
	`Radiobutton (15,20) "Gramm" Name="einheit.g"`
	`Radiobutton (16,20) "Kilo" Name="einheit.kg"`
	
	• Durch den Parameter `Name="iname ..."` erhält das Eingabefeld einen internen

Namen. Der Wert kann durch `[iname]` angesprochen werden

- Bei Checkbox und Radiobutton sind `" "` und `"X"` mögliche Werte.
- Der Name eines Radiobutton muß aus dem Namen der Gruppe sowie, durch Punkt getrennt, aus dem Namen des Buttons bestehen, also z.B. `"einheit.kg"`. Der Name der Gruppe bestimmt, welche Radiobuttons eine logische Einheit bilden. Wenn ein Button aus der Gruppe aktiviert wird, werden automatisch alle anderen deaktiviert.
- F1 bei eigenen Feldern ist nicht unterstützt. Man kann jedoch einzeilige Tips als Quickinfo anzeigen lassen. Empfehlenswert ist eventuell auch das Einblenden eines Hilfesymbols mit `Image`, und die Anzeige einer .doc oder .html Dokumentation bei Klick auf das Hilfesymbol (siehe `Image`).
- F4 bei eigenen Feldern ist nicht unterstützt. Der InputAssisant zeigt jedoch bei Klick auf rechte Maustaste sowohl die Eingabehistorie als auch, falls vorhanden (Generierung), die lokale Wertehilfe an
- Vor SAP Rel. 4.0 ist es nötig, die aktuelle Transaktion durch die Anweisung `Transaction` im GuiXT Script anzugeben, andernfalls wird im Fehlerfall auf das letzte Menü statt auf das Grundbild der Transaktion verzweigt. Bei Start aus einem Menübild ist die Angabe einer Transaktion nicht erforderlich. Beispiel:

```
Transaction "FB01"
```

Pushbutton

```
Pushbutton (row,col) "Text" "fcode" Process="..."
```

`"Process="..."` verweist auf eine Datei, in der das InputScript abgelegt ist. Die "fcode"-Angabe kann entfallen, falls nur "Enter" ausgelöst werden soll.
Beispiele:

```
Pushbutton (10,40) "Material anlegen"
Process="CreateMaterialSimple.txt"
Pushbutton (toolbar) "Material anlegen" "/nmm01"
Process="CreateMaterialSimple.txt"
```

- Durch `using` ist die Parametrisierung des InputScript möglich. Hierzu werden im Anschluß an die Pushbutton-Anweisung die zu übergebenden Parameter aufgeführt Die Namen der Parameter sind im InputScript definiert.
- Beispiel:

```
Pushbutton (toolbar) "Material anlegen"
Process="CreateMaterialSimple.txt"
     using Grunddaten      = [nurGD]
     using MEStueck        = [einheit.st]
     using MEKilo          = [einheit.kg]
     using Lagerort        = "0001"
```

6.7. Lokale Wertehile und Matchcode für eigene Eingabefelder

Die Standard Hilfefunktion von R/3 (F4) steht für neue Eingabefelder nicht automatisch zur Verfügung.

Sie haben drei Möglichkeiten, hier Abhilfe zu schaffen:

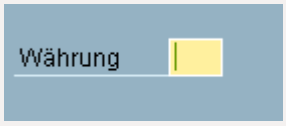
- Verwenden Sie Radiobuttons (Auswahlknöpfe) anstelle eines Eingabefeldes
- Definieren Sie die lokale Wertehilfe für ein neues Eingabefeld
- Definieren Sie eine Verbindung zu den Standard R/3 Hilfetexten (Matchcode)

Radiobuttons anstelle eines Eingabefeldes verwenden

Dies ist sinnvoll bei einer begrenzten kleinen Anzahl von Eingabemöglichkeiten.

Vorgehensweise: Anstelle der Definition Ihres neuen Eingabefeldes ...

```
InputField (10,1) "Währung" (10,12) Size="3" Name="Vwaer"
```



... verwenden Sie eine Gruppe von Radiobuttons für die gleiche Variable:

```
Offset (10,1)
Box (0,0) (5,40)  "Währung"
Radiobutton (1,1) "USD US Dollar"          Name="Vwaer" Value="USD"
Radiobutton (2,1) "EUR Euro"               Name="Vwaer" Value="EUR"
Radiobutton (3,1) "GBP Britisches Pfund"   Name="Vwaer" Value="GBP"
Radiobutton (4,1) "MXN Mexikanische Pesos" Name="Vwaer" Value="MXN"
```

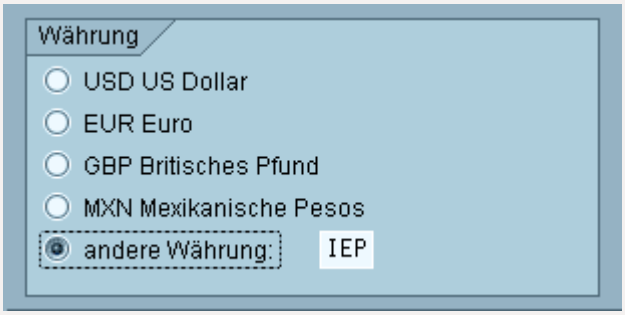
Sie können auch einen Default zuweisen, indem Sie die Variable V[MeineWährung] mit einem Wert belegen:

```
Offset (10,1)
Box (0,0) (5,40) "Währung"

// "USD" als Default definieren wenn V[MeineWährung] nicht belegt ist
if not V[Vwaer]
  Set V[Vwaer] "USD"
```

```
endif

Radiobutton (1,1) "USD US Dollar"          Name="Vwaer" Value="USD"
Radiobutton (2,1) "EUR Euro"               Name="Vwaer" Value="EUR"
Radiobutton (3,1) "GBP Britisches Pfund"   Name="Vwaer" Value="GBP"
Radiobutton (4,1) "MXN Mexikanische Pesos" Name="Vwaer" Value="MXN"
```

Weiterhin können Sie auch Radiobuttons mit einem Eingabefeld kombinieren:

```
offset (10,1)
box (0,0) (6,40) "Währung"

// "USD" als Default definieren wenn V[Vwaer] nicht belegt ist

if not V[Vwaer]
  Set V[Vwaer] "USD"
endif

Radiobutton (1,1) "USD US Dollar"          Name="Vwaer" Value="USD"
Radiobutton (2,1) "EUR Euro"               Name="Vwaer" Value="EUR"
Radiobutton (3,1) "GBP Britisches Pfund"   Name="Vwaer" Value="GBP"
Radiobutton (4,1) "MXN Mexikanische Pesos" Name="Vwaer" Value="MXN"
radiobutton (5,1) "andere Währung:"        Name="Vwaer" Value="XXX"
inputfield (5,20) name="Vandere" size="3" "-nolabel"
```



In dem zugeordneten InputScript benötigen Sie dann eine IF Anweisung für den Fall der "Anderen Währung" :

```
// Wurde "Andere Währung" ausgewählt?
if V[Vwaer=XXX]
  Set V[Vdiewaer] "&V[Vandere]"
else
  Set V[Vdiewaer] "&V[Vwaer]"
endif
```

Die lokale Wertehilfe für ein neues Eingabefeld definieren

In diesem Fall verwenden Sie die normale InputField Anweisung:

```
InputField (10,1) "Währung" (10,12) Size="3" Name="Vwaer"
```

Zusätzlich können Sie dem Eingabefeld einen Domänenamen zuweisen in der Datei **domText.D.txt,** die Sie in Ihrem Domänenverzeichnis finden: (vgl. Profile)

Währung: Waers

Wenn Sie nicht mit einem Standard Domänenamen arbeiten möchten, verwenden Sie stattdessen die Datei **domTextScreen.D.txt:**

SAPMC29C.0584.Währung: Meinewaer

Hier ist **SAPMC29C** der Programmname, **0584** die Screen Nummer, **Währung** der Name des Eingabefeldes, und **Meinewaer** ein Domänename, den Sie für Ihr Feld erfunden haben.

Danach erstellen Sie die Datei für die lokale Wertehilfe, entweder

dom.D.Waers.txt

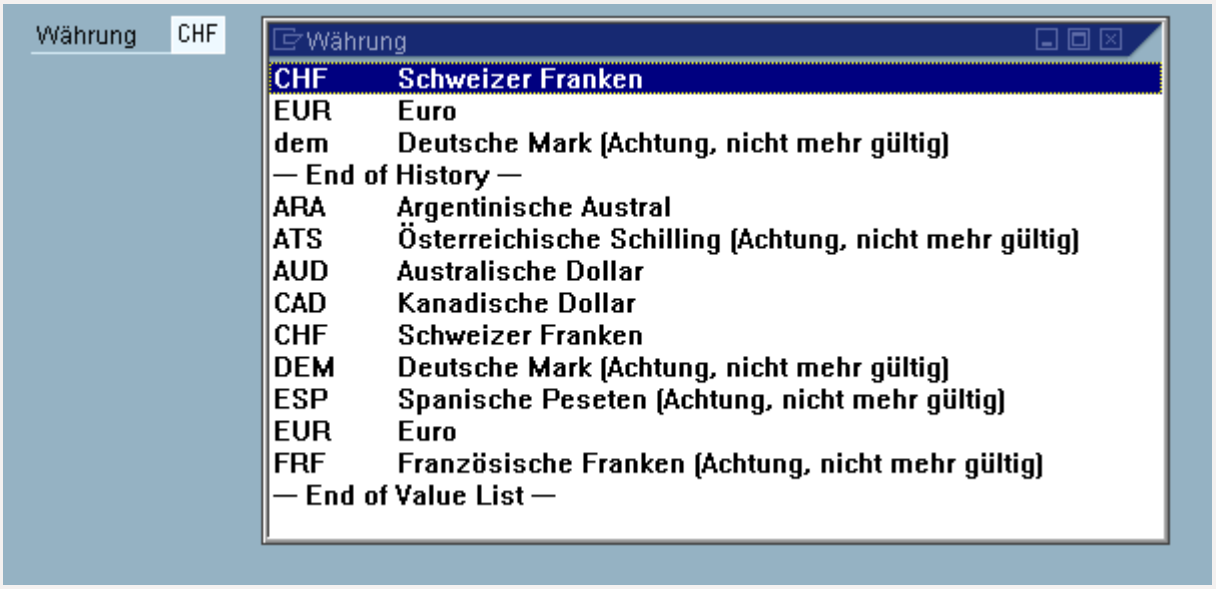
oder

dom.D.Meinewaer.txt

die etwa so wie in diesem Beispiel aussehen könnte:

ARA //Argentinische Austral
ATS //Österreichische Schilling (Achtung, nicht mehr gültig)
AUD //Australische Dollar
CAD //Kanadische Dollar
CHF //Schw eizer Franken

Bei rechtem Mausklick auf das Eingabefeld werden die Werte angezeigt, die in der Vergangenheit eingegeben wurden (History), zusammen mit Erklärungstexten und gefolgt von einer Liste aller möglichen Währungen. Diese Liste ist frei editierbar, wie Sie an unserem Beispiel sehen können.


Währung
CHF
Währung
CHF Schweizer Franken
EUR Euro
dem Deutsche Mark (Achtung, nicht mehr gültig)
— End of History —
ARA Argentinische Austral
ATS Österreichische Schilling (Achtung, nicht mehr gültig)
AUD Australische Dollar
CAD Kanadische Dollar
CHF Schweizer Franken
DEM Deutsche Mark (Achtung, nicht mehr gültig)
ESP Spanische Peseten (Achtung, nicht mehr gültig)
EUR Euro
FRF Französische Franken (Achtung, nicht mehr gültig)
— End of Value List —

Die lokale Werteliste ist sehr schnell. Sie bietet die zuletzt getätigten Eingaben zusammen mit Erklärungstexten und allen Möglichkeiten zur Auswahl. Zu Details, z.B. wie Sie die lokale Wertehilfe generieren, vergleichen Sie bitte die entsprechende Seite zum InputAssistant.

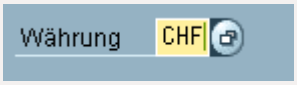
Eine Verbindung zu den R/3 Standard Hilfetexten definieren

Für Eingabefelder für Kunden, Bestellungen, Materialien ... können Sie weder Radiobuttons noch die lokale Wertehilfe verwenden, da das Eingabefeld sich auf eine relativ große und dynamische

Wertemenge bezieht.

Wenn Sie Ihr Eingabefeld mit dem R/3 Repository verbinden, indem Sie den technischen Namen des geeigneten R/3 Data Dictionary Feldes angeben, dann verarbeitet GuiXT automatisch die korrespondierenden Hilfetexte .

```
InputField (10,1) "Währung" (10,12) Size="3" Name="Vwaer" Techname="BKPF-WAERS"
```



Wäh...	Langtext
ADP	Andorianische Peseta
AED	Vereinigte Arabische Emirate Dirham
AFA	Afghani
ALL	Albanische Lek
AMD	Armenische Dram
ANG	Antilianische Gulden
AOK	Angolesische Kwanza
ARA	Argentinische Austral
ATS	Österreichische Schilling
AUD	Australien Dollar
AWG	Arubanische Gulden
AZM	Aserbaidschanische Manat
BAD	Bosnien-Herzegowinischer Dinar
BBD	Barbados-Dollar
BDT	Bangladesch Taka
BEF	Belgische Franken
BGL	Bulgarische Lew
BHD	Bahrein-Dinar
BIF	Burundi Frank
BMD	Bermuda Dollar
BND	Brunei Dollar
BOB	Boliviano
BRC	Brasilianische Real
BRL	Brasilianische Real
BSD	Bahama Dollar
BWP	Botswanische Pula
BYR	Weissrussische Rubel
BZD	Belize-Dollar
CAD	Kanadische Dollar
CHF	Schweizerische Franken
CLP	Chilenische Pesos
COP	Kolumbianische Peso
CRC	Costa-Rica-Colon

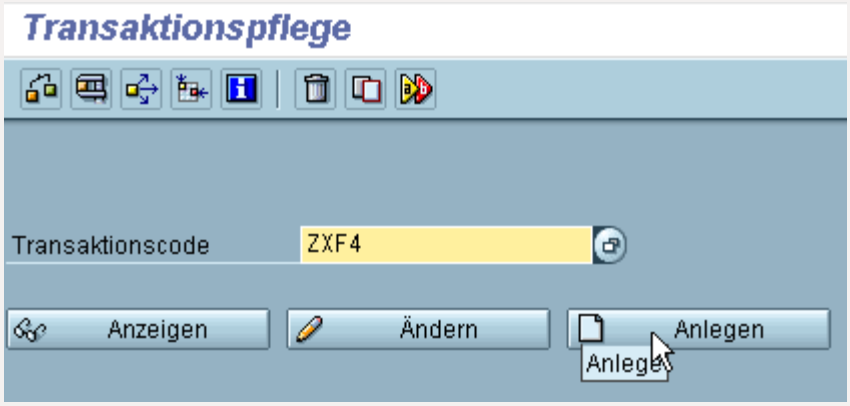
Das funktioniert nur dann, wenn Sie die folgenden R/3 Objekte in Ihrem System installiert haben:

- ABAP program ZGUIXTF4 (wird bei der Installation von GuiXT in das Installationsverzeichnis gestellt)

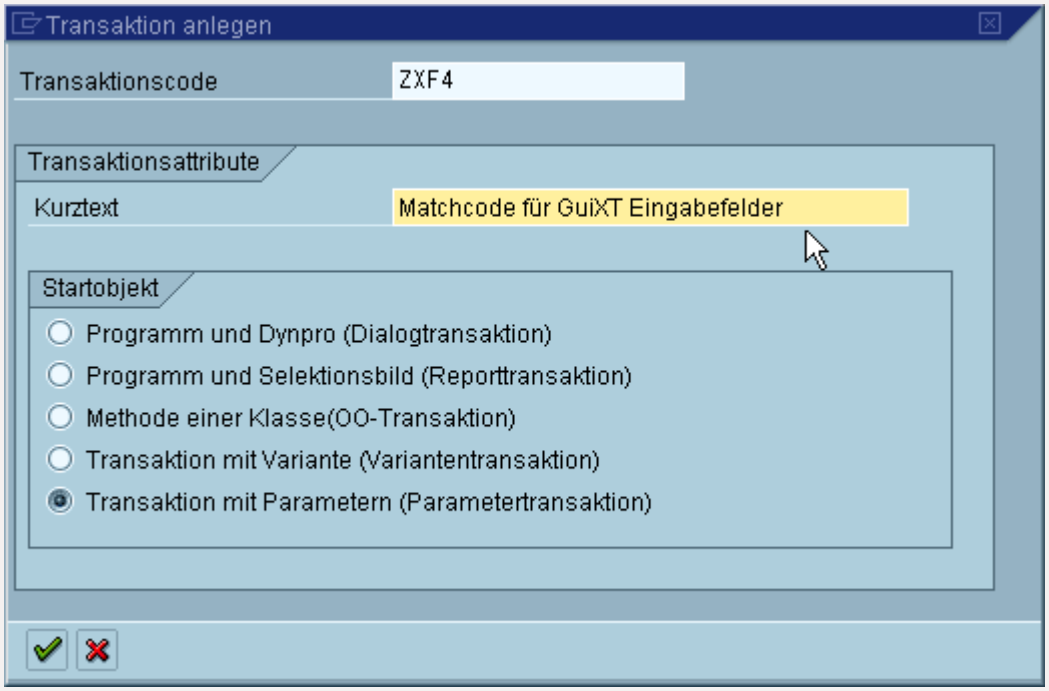
- Transaktionscode ZXF4

Um den Transaktionscode ZXF4 zu erstellen, gehen Sie folgendermaßen vor:

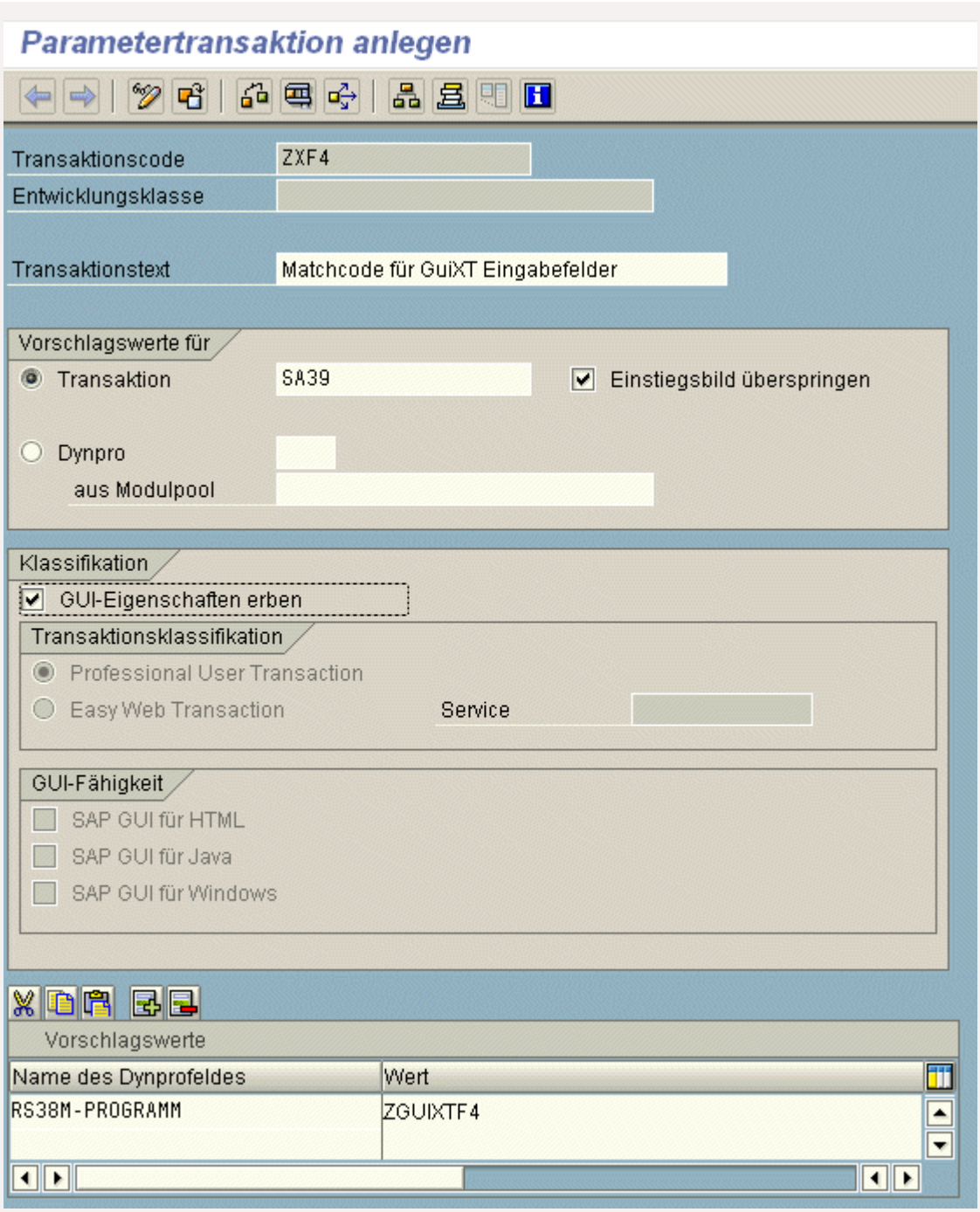
Starten Sie die Transaktion **SE93**, geben Sie den neuen Transaktionsnamen ZXF4 ein und drücken Sie den **Anlegen** Druckknopf:



Definieren Sie ihn als **Parametertransaktion**:

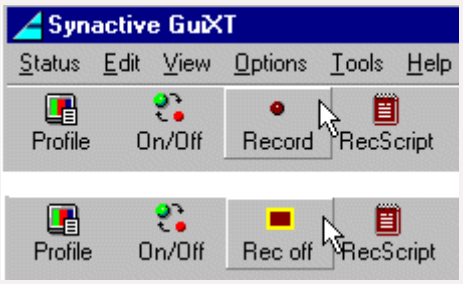


Die neue Transaktion ist eine Parametertransaktion, die ZGUIXTF4 aufruft. Sie müssen sodann den Transaktionscode **SA39** (internal transaction) eingeben, markieren Sie die Checkbox "Einstiegsbild überspringen" und geben Sie den Namen des Dynprofeldes **RS38M-PROGRAMM** mit dem Wert **ZGUIXTF4** an:



Zum Schluß drücken Sie den **Sichern** Button und geben eine passende Entwicklungsklasse an, oder, wenn kein Transport in andere R/3 Systeme geplant sind, geben Sie es als lokales Objekt an.

6.8. InputRecording

Funktion des Input Recording	Das Input Recording zeichnet alle Eingabewerte und Aktionen auf und erzeugt daraus eine InputScript Datei im Textformat (.txt).
Aufruf des Input Recording	Sie können die Funktion durch Klicken auf Options->Input Recording aktivieren und mit Edit->Recorded InputScript das aufgezeichnete Script editieren. Alternativ stehen in der Werkzeugleise auch Buttons zur Verfügung:  **Das Input Recording kann durch einen Ein/Aus Schalter gesteuert werden. Mit dem Button RecScript wird die aufgezeichnete Datei im Editor aufgerufen.**
Verwendung der aufgezeichneten InputScript Datei	Eine Aufzeichnung der Eingabewerte und Aktionen erleichtert Ihnen die Arbeit bei der Erstellung von InputScripts, beispielsweise um Einbildtransaktionen zu erstellen. Die Aufzeichnungen können Sie im Editor bearbeiten und unter einem neuen Namen abspeichern. Die durch den Input Recorder erzeugte Datei InputScript.txt wird durch jeden neuen Aufzeichnungsvorgang überschrieben. Meist ersetzt man die eingegebenen Werte durch Prameter oder durch die Werte von InputField Variablen. DIe generierten "SetCursor" Anweisungen können SIe meist streichen, es sei denn, Sie lösen eine Funktion aus, bei der der Cursor auf ein bestimmtes Feld zeigen muss (Doppelklcik bzw. /2).

Beispiel einer Aufzeichnung durch den Input Recorder (FB01)	**Das auf dieser Basis entwickelte InputScript**

`Screen SAPMF05A.0300`
`  SetCursor [Konto]`
`  Set F[Betrag] "1000"`
`  Set F[Bschl] "50"`
`  Set F[Konto] "113101"`
`  Enter`
 	
`Screen SAPMF05A.0300`
`  SetCursor [Betrag]`
`  Set F[Betrag] "1000"`
`  Enter "/11"`
	`Screen SAPMF05A.0300`
	`  Set F[Betrag]  "&V[Betrag1]"`
	`  Set F[Bschl]  "&V[Beschl2]"`
	`  Set F[Konto]  "&V[Konto2]"`
	`  Enter`
	
	`Screen SAPMF05A.0300`
	`  Set F[Betrag]  "&V[Betrag2]"`
	`  Enter "/11"`

7. Die Befehle im Detail (InputAssistant)

7.1. Appendfile

Funktion	Mit der **AppendFile**-Anweisung können Sie einen Satz in eine Datei schreiben, z.B. um Daten aus SAP ERP zur weiteren Verarbeitung in Excel zur Verfügung zu stellen. Es ist auch möglich, eine andere Datei unter Ersetzung von Variablen in die Datei zu übertragen.
Beispiel	`AppendFile "X:GuiXTFiles\MatDes.txt" s_matnr s_text s_unit s_price` Der Inhalt der Variablen `&[s_matnr] &[s_text] &[s_unit] &[s_price]` wird zu einem einzigen Datensatz zusammengefügt, indem zwischen die einzelnen Felder das bei `OpenFile` angegebene Trennzeichen (`Delimiter=`) eingefügt wird (Default-Trennzeichen ist das Tabulatorzeichen). Ist etwa `&[s_matnr]` = 10000120 `&[s_text]` = Schraube `&[s_unit]` = ST `&[s_price]` = 0,02 und das Trennzeichen ein ';', dann wird der Satz 1000001;20;Schraube;ST;0,02 in die Datei eingefügt.

Format	`AppendFile "filename" var1 var2 var3 ..."`
	
	Die Angabe von maximal 60 Variablen ist möglich (vor GuiXT Version 2009 Q1 2: 20 Variablen).
	
	`AppendFile "filename" template="templatefile"`
	
	Bei der Übertragung der `template` Datei werden alle GuiXT-Variablen der Form **&[...]** durch ihre aktuellen Werte ersetzt.

Tipps & Tricks	Ein ausführliches Beispiel finden Sie in www.synactive.com unter "Dokumentation - Special Topics".

7.2. ApplyGuiScript

Funktion	Mit der **ApplyGuiScript**-Anweisung können Sie die den aktuellen Screen durch ein VBScript weiter bearbeiten (ab SAP GUI 6.20, "SAP GUI Scripting API for Windows"). Hierdurch ist insbesondere der Zugriff auf spezielle Controls (z.B. grid-Control) möglich.
Beispiel	`ApplyGuiScript "activate_document_link.vbs"`

Format	`ApplyGuiScript "Dateiname"`
	`"Dateiname"` ist der Name einer .vbs-Datei. Sie kann mit oder ohne Pfad (aktuelles Script-Verzeichnis) angegeben werden. Auch Dateien aus dem SAP Web Repository sind möglich.
	`ApplyGuiScript template="templatedateiname"`
	`"templatedateiname"` ist der Name einer .vbs-Datei. Alle GuiXT-Variablen "&V[....]", "&F[...]", ... werden vor der Ausführung des Scripts durch die aktuellen Werte ersetzt.

Tipps & Tricks	Weitere Informationen und Beispiele finden Sie in den "Special Topics" im Artikel "SAP GUI Scripting und GuiXT".

7.3. Call

Funktion	Mit der `Call`-Anweisung können Sie aus einem GuiXT Skript oder einem InputScript heraus eine Funktion aufrufen. Die Funktion kann entweder lokal (am Frontend) als dll-Funktion vorliegen (z.B. in VC++ implementiert), oder Sie rufen eine in ABAP geschriebene Funktion auf dem Applikationsserver auf. In diesem Fall wird die Remote Function Call Technologie (RFC) zum Aufruf des Funktionsbausteins eingesetzt. Zu beachten: Die `Call`-Anweisung über RFC setzt die Komponente "InputAssistant" voraus.

Beispiele	`Call "ImgName" dll="guiexits" In="&F[Material]" Out="Datei"`
	`Call "ImgName" In="&F[Material]" Out="MatFileName"`
	Im ersten Fall wird lokal die dll-Funktion `ImgName`aufgerufen. Im zweiten Fall wird über RFC der Funktionsbaustein `ImgName`im SAP aufgerufen.

Format	**RFC** **Aufruf**
	`Call "Funktionsname" In.Name1="Par1" In.Name2="Par2" ... Out.Name1="Par1" Out.Name2="Par2"... Table.Name1="tab1" Table.Name2="tab2" ...`
	`Call "Funktionsname" Destination="dest" In.Name1="Par1" In.Name2="Par2" ... Out.Name1="Par1" Out.Name2="Par2"... Table.Name1="tab1" Table.Name2="tab2" ... ...`
	dll Aufruf
	`Call "Funktionsname" dll="dllname" In="Par1" In="Par2" ... Out="Par1" Out="Par2" ...`
	Zu beachten: Zur Zeit ist die Gesamtzahl der Parameter (`In` + `Out` + `Table`) auf

	maximal 20 begrenzt.	

Zusatzoptionen	-try	Bei RFC, Option bitte unmittelbar hinter dem Namen des Funktionsbausteins angeben.
		
		Falls der Funktionsbaustein mit einer Exception abbricht, wird keine Fehlermeldung an den Benutzer ausgegeben. Stattdessen wird in der Systemvariablen V[_exception] der Name der ausgelösten Exception zur Verfügung gestellt. Durch `if Q[ok]` kann nach `Call`abgefragt werden, ob der Funktionsbaustein normal beendet wurde. Beispiel:
		
		`Call "RPY_TABLE_READ" -try in.TABLE_NAME="&V[structid]" ..`
		`if not Q[OK]`
		`Return "E: Struktur &V[structid] nicht im Data Dictionary gefunden" -statusline`
		`endif`

	-currentuser	Bei RFC, Option bitte unmittelbar hinter dem Namen des Funktionsbausteins angeben. Der Aufruf wird mit dem gerade angemeldeten Benutzer durchgeführt, nicht mit dem im GuiXT Profile angegebenen RFC-Benutzer. **Bitte beachten:** Diese Option ist zur Zeit nicht bei Single Sign On unterstützt, sondern nur bei SAP-Anmeldung mit Benutzername und Passwort. `Call "RPY_TABLE_READ" -currentuser in.TABLE_NAME="&V[structid]" ..`
	cache=	Bei RFC, Option bitte unmittelbar hinter dem Namen des Funktionsbausteins angeben. Möglich sind die Angaben `cache="transaction"` und `cache="session"`. Wirkung: Es wird vor Ausführen des RFC geprüft, ob bereits ein Aufruf des Funktionsbausteins mit den gleichen Eingaben stattgefunden hat. Falls ja, werden statt des RFC Aufrufs die Rückgabewerte aus einem internen Cache gelesen und zurückgeliefert. Falls nein, wird der RFC ausgeführt; anschliessend werden die zurückgelieferten Daten im Cache vermerkt. Als Eingabewerte zählen die mit in.xxx= angegebenen Werte plus alle Tabelleninhalte table.xxx=. Falls eine Tabelle "abc" von dem Funktionsbaustein nicht verarbeitet, sondern nur zurückgeliefert wird, ist es sinnvoll, sie vor dem Aufruf durch `Set text[abc]""` zu löschen, damit ihr Inhalt nicht zu den Cache-Eingabewerten dazugenommen wird. Das würde zwar nicht zu fehlerhaften Daten

führen, die Qualität des Cache aber verschlechtern.

Als Ausgabewerte zählen die mit out.xxx= angegebenen Parameter plus alle Tabelleninhalte table.xxx=.

Die Gesamtlänge aller Eingabewerte inklusiv der Parameternamen ist im Cache auf 8000 Zeichen beschränkt. Bei längeren Werten (insgesamt) wird die `cache=` Angabe ignoriert.

Bei Angabe von `cache="transaction"` wird ein Cache verwendet, der pro Neuaufruf einer Transaktion zurückgesetzt wird. Das heisst, wenn der Benutzer eine Transaktion neu mit /N... oder aus dem Menü aufruft, werden alle Daten frisch gelesen. Bei Angabe von `cache="session"` wird dagegen ein separater Cache verwendet, der für alle Transaktionsaufrufe innerhalb eines Modus gültig ist.

Der Session-Cache ist sinnvoll für Werte wie Benutzereinstellungen oder Customizing-Daten, während der Transaktionscache für Anwendungsdaten wie Kundenadressen verwendet werden kann, die sich im Prinzip jederzeit ändern können, die aber während eines Transaktionsaufrufs nicht ständig nachgelesen werden müssen.

Für ein eigenes Rücksetzen des Cache können Sie `ClearCallCache cache="transaction"` und `ClearCallCache cache="session"` verwenden. In der Einstiegsmaske einer Transaktion kann es sinnvoll sein, durch `ClearCallCache cache="transaction"` den Transaktionscache zurückzusetzen, falls der Benutzer auch z.B. durch F3 oder F12 aus einem anderen Maske der Transaktion auf das Einstiegsbild gelangen kann, ohne die Transaktion neu aufzurufen. In diesem Fall beginnt nämlich die SAP-Anwendung technisch keine neue Transaktion, sodass der bisherige Transaktionscache noch gültig ist.

Ausgabeparameter

Die Ausgabeparameter des `Call` können in der Form `&[name]` in allen nachfolgenden Skriptzeilen angesprochen werden. Beispiel:

```
Call "ImgName" In="&[Material]" Out="MatFileName"
Image (16,81) (24,100) "&[MatFileName]" Start="&[MatFileName]" -
NoStretch
```

dll-Aufruf

Die Funktion wird bei z.B. 2 IN und 3 OUT-Parametern vom Typ

__declspec(dllexport) int funcname(char* p1, char*p2, char* p3, char* p4, char* p5)

definiert. Alle Parameter (`In=` und `Out=`) werden in der angegebenen Reihenfolge übergeben. Die maximale Länge jedes einzelnen Strings beträgt 4000 Zeichen. Bitte achten Sie darauf, in dem Skript genügend viele Parameter anzugeben, sonst greift die dll-Funktion auf eine ungültige Adresse zu.

Einige nützliche dll-Funktionen können Sie sich auf unserer dll-download-Seite

besorgen

RFC Aufruf

Parameterübergabe

Es gibt hierfür 2 Varianten. Entweder arbeiten Sie mit Positionsparametern (ohne Nennung eines Namens):

```
Call "funktionsname" In="Par1" In="Par2" ... Out="Par1" Out="Par2"
...
```

In diesem Fall müssen die Importing-Parameter des Funktionsbausteins mit In1, In2, ... benannt sein, die Exporting-Parameter mit Out1, Out2,... (SAP Funktionsbibliothek, Transaktion SE37). Tables-Parameter sind dann nicht möglich.

Oder Sie arbeiten mit benannten Parametern:

```
Call "funktionsname" In.Name1="Par1" In.Name2="Par2" ...
Out.Name1="Par1" Out.Name2="Par2" ... Table.Name1="tab1"
Table.Name2="tab2"
```

In diesem Fall können die Namen der Parameter Name1, Name2, ... beliebig sein.

Restriktionen

- Die Schnittstelle darf lediglich aus Parametern vom Typ C=Character bestehen, Maximallänge ist 4000. Sie können zum Beispiel als Referenzfeld jeweils das Feld SY-LISEL angeben
- EXCEPTIONS können nicht im Skript abgefangen werden. Es erfolgt in diesem Fall eine entsprechende Nachricht, und der Call wird übergangen
- Dialoge innerhalb des Funktionsbausteins sind nur möglich, wenn Sie den Funktionsaufruf "**Call Function SYSTEM_ATTACH_GUI.**" in Ihrem Funktionsbaustein verwenden. Falls bereits in Ihrem SAP-System vorhanden, stattdessen bitte die neueren Bausteine "**Call Function SYSTEM_PREPARE_ATTACH_GUI.**" und "**Call Function SYSTEM_FINISH_ATTACH_GUI**." verwenden.
- Die übergebenen Tabellen dürfen nur zeichenartige Felder enthalten, keine gepackten Werte oder Integer
- Die Breite der Tabelle ist auf 256 Zeichen beschränkt (beliebig viele Zeilen sind möglich). Sie können in einem Call pro Tabelle die benötigte Breite aber selbst auf einen Wert zwischen 1 und 32000 setzen durch den Zusatz (width:xxxx) hinter dem Tabellennamen:
  `Table.Name1(width:4000)="tab1"`

Aus Ihrem eigenen Funktionsbaustein können selbstverständlich beliebige andere Funktionsbausteine aufgerufen werden.

Handhabung von Tabellen

- Tabellen werden in InputScripts wie Langtexte behandelt. Sie können z.B. die

Anweisungen `TextBox` und `CopyText` benutzen.

- Sie werden in beiden Richtungen übertragen
- Es gibt keine Unterstützung von Data Dictionary Strukturen in InputScripts. Um einzelne Felder aus einer strukturierten Tabellenzeile anzusprechen, müssen Sie die Substring-Notation verwenden.

Beispiel: Aufruf eines BAPI als RFC
Sie benutzen die Methode "Get_Detail" des SAP Businessobjekts "User", um die Benutzergruppe (Informationen aus Benutzerstammsatz) zu lesen:

```
Call "BAPI_USER_GET_DETAIL" In.Username="&[_user]"
Out.LogonData="UserLogonData"
Set V[UserGroup] "&[UserLogonData](18-29)"
```

Danach steht in der Variablen `&[UserGroup]` die Benutzergruppe zur Verfügung.

Erläuterung (vergleichen Sie die Schnittstellendefinition in Transaktion SE37):

- Den Importing-Parameter `Username` besetzen Sie mit der Systemvariable `&[_user]`
- Zurückgeliefert wird der Exporting-Parameter `LogonData` im Parameter `&[UserLogonData]`
- Laut Definition der Struktur `BapiLogonD` steht in den Stellen 18-29 die Benutzergruppe

Beispiel: RFC-Aufruf aus einem InputScript

Aufruf in anderen Systemen

Es ist möglich, Funktionen in anderen SAP Systemen aufzurufen. Verwenden Sie hierzu den Parameter `Destination=`... Die dort angegebene Destination muss in der Datei **saprfc.ini** beschrieben sein, vergleichen Sie bitte die SAP-Dokumentation zum RFC. Die Datei saprfc.ini liegt entweder in dem SAP- Workdirectory oder eine Umgebungsvariable RFC_INI benennt die Datei.

Hierdurch können Sie sehr leicht Informationen aus weiteren Systemen in die SAP Bilder integrieren.

Tipps & Tricks	• Falls Sie zur Implementierung der dll statt VC++ andere Programmiersprachen (z.B. Visual Basic) verwenden wollen, sind eventuell zusätzliche Tools zur Erstellung der dll nötig. Bitte fragen Sie gegebenenfalls bei dem Hersteller der entsprechenden Programmiersprache nach. • Zum Austesten des ABAP Funktionsbausteins empfiehlt sich die Testumgebung der ABAP Workbench. Für ein Debugging bei Aufruf aus dem Skript heraus müssen Sie den Aufruf "**Call Function SYSTEM_ATTACH_GUI.**" vor dem ersten Break-Point verwenden. Falls bereits in Ihrem SAP-System vorhanden, stattdessen bitte die neueren Bausteine "**Call Function SYSTEM_PREPARE_ATTACH_GUI.**" und "**Call Function SYSTEM_FINISH_ATTACH_GUI.**" verwenden. • Der für den RFC verwendete RFC Benutzername und das Passwort sind im GuiXT-Profile hinterlegt (Passwort ist verschlüsselt). Standardmäßig wird der Benutzer SAPCPIC mit seinem Standardpasswort benutzt.

7.4. Checkbox

Funktion	Mit der `CheckBox`-Anweisung können Sie eigene Ankreuzfelder definieren. Der dort eingetragene Wert (`"X"` oder `" "`) kann in einem InputScript verwendet werden.
Beispiel	`CheckBox (10,1)"Umweltrelevant" Name="Umwelt"` Es wird ein Ankreuzfeld auf Position `(10,1)` definiert. Der interne Name ist Umwelt, d.h. in einem InputScript kann der eingegebene Wert als `&[Umwelt]` abgerufen werden.
Format	`CheckBox (Zeile,Spalte) "text" Name="..."`
Zusatzoptionen	`Default="..."` Angabe eines Defaultwerts, nur `"X"` oder `" "` sind möglich `-readonly` Nur Ausgabe
Tipps & Tricks	• Sie können bei dem Feldtext eine Quickinfo hinterlegen, indem Sie den Text in der SAP-Konvention für Quickinfos aufbauen, z.B.: `"@0L\QAchtung: Sicherheitsdatenblätter erforderlich@Umweltrelevant"` • Sie können eine Checkbox auch durch ein InputScript mit dem Befehl `set` ankreuzen. Siehe dort unter Tipps & Tricks.

7.5. Checkdate

Funktion	Mit der `Checkdate`-Anweisung können Sie prüfen, ob eine Zeichenreihe ein gültiges Datum darstellt. Dabei wird für das Datumsformat der im SAP-System abgelegte Benutzerwert verwendet.

	Durch `if Q[ok]` können Sie anschliessend abfragen, ob es sich um ein gültiges Datum handelt. **Bitte beachten**: Beim ersten Verwenden dieser Anweisung wird das Datumsformat des Benutzers über einen Remote Function Call (RFC) eingelesen. Es ist daher ein gültiger RFC Benutzer im GuiXT-Profile erforderlich.

Beispiel	`CheckDate "&V[mydate]"`
	`if not Q[ok]`
	`Message "E: Datumsangabe &V[mydate] nicht korrekt"-statusline`
	`Leave`
	`endif`

Format	`CheckDate "datum"`
Zusatzoptionen	`set=` Angabe einer Variablen, in die der Datumswert in vollständiger 10-stelliger Form gestellt wird. Bei einer falschen Datumsangabe erhält die Variable den Wert "". Beispiel: `CheckDate "1.6." set="cdate"` Danach hat die Variable V[cdate] den Wert "01.06.2011", falls das Benutzerformat "dd.mm.yyyy" ist und das aktuelle Jahr 2011. Bitte beachten Sie, dass die Option -date bei InputField bereits die vollständige Datumsform herstellt, falls es sich um ein gültiges Datum handelt. Bei ungültigem Datumsformat bleibt der Eingabewert unverändert, es erfolgt aber keine automatische Fehlermeldung.

7.6. ClearCallCache

Funktion	Mit der `ClearCallCache`-Anweisung können Sie den bei `Call` verwendeten Cache zurücksetzen.
Beispiel	`ClearCallCachecache="transaction"`

Format	`ClearCallCache`
	`Setzt sowohl den Transaktions-Cache als auch den Session-Cache zurück.`
	`ClearCallCachecache="transaction"`
	Setzt den Transaktions-Cache zurück.
	`ClearCallCachecache="session"`
	Setzt den Session-Cache zurück.

7.7. Closefile

Funktion	Mit der `CloseFile`-Anweisung können Sie eine Datei schließen.

Beispiel	`CloseFile "X:GuiXTFiles\MatDes.txt"` Die Datei wird geschlossen. Die nächste `ReadFile` Anweisung liest dann wieder den ersten Satz der Datei.
Format	`CloseFile "filename"`
Tips &Tricks	• Das Öffnen oder Schließen einer Datei bezieht sich auf den jeweiligen SAP Modus • Ein ausführliches Beispiel finden Sie im Special Topic "Die Behandlung von Dateien in InputScripts – Dateien lesen und schreiben während automatisierter Transaktionsverarbeitung".

7.8. Column

Funktion	Mit der `Column`-Anweisung können Sie Spalten in einem Table Control hinzufügen.
Beispiel	`Column "Beschreibung" size=30 name="descr" position=5 -readonly` Hierdurch wird eine neue Spalte mit der Überschrift "Beschreibung" hinzugefügt. Die Werte in der Spalte werden aus den Variablen V[descr.1], V[descr.2],... entnommen.
Format	`Column "Spaltentitel" size=x name="varname" (options)` Eine neue Spalte mit Überschrift "Spaltentitel" wird hinzugefügt. Die Spaltenbreite (und auch maximale Eingabelänge) ist x. Die Variablen V[varname.1], ... werden den Spaltenwerten zugeordnet (Eingabe und Ausgabe), wobei der Index 1,2,3,,, die absolute Zeilennummer innerhalb des gesamten Table Control ist, nicht nur innerhalb des gerade sichtbaren Bereichs. Es ist auch möglich, statt der Zeilennummer die Werte einer anderen Spalte als Index zu benutzen; siehe Option `keycolumn=`.

Optionen

`table=`	Die Tabelle T[tabname] kann explizit angegeben werden, was im Fall mehrerer Tabellen auf einem Screen auch erforderlich ist.
`position=`	Spaltennummer in der Anzeige
`keycolumn=`	Statt die Zeilennummer als Index zu verwenden, können Sie eine andere Spalte angeben, deren Werte statt 1,2,3... als Index genommen werden. Zum Beispiel können Sie durch `keycolumn=[Pos]` eine Spalte angeben, in der das System die jeweilige Positionsnummer innerhalb eines Auftrags führt. Falls in Ihrer neuen Spalte auch Eingabe möglich ist, und der Benutzer nachträglich Zeilen löschen oder hinzufügen kann, ist die absolute Zeilennummer kein brauchbarer Index, da er sich bei Löschen oder Hinzufügen von Zeilen ändert. Ein Beispiel für die Verwendung finden Sie in den Tutorials zum InputAssistant.

	-numerical	Nur Zifferneingabe
	-uppercase	Nur Grossbuchstaben
	-intensified	Der Wert wird farblich hervorgehoben (Standard: rot)
	-alignright	Der Wert wird rechtsbündig ausgegeben
	-readonly	Nur Ausgabe
	-proportionalfont	Ausgabe mit Proportionalfont
	-checkbox	In der Spalte wird jeweils eine Checkbox angezeigt. Die zugeordneten Variablenwerte sind "X" und " "
	-pushbutton	Drucktaste
	process=	Nur für Drucktasten: Angabe des InputScripts, das prozessiert werden soll. Die Zeilennummer in der Tabellenanzeige steht in V[_tabrow].
	label=	Fester Drucktastentext. Optional mit Icone und Tooltip, Syntax wie bei Pushbutton: "@id\Qtooltip\text". Falls label= nicht angegeben ist, wird der Inhalt der Variablen (name=) pro Zeile als Drucktastentext interpretiert. Er kann dann pro Zeile unterschiedlich sein.
	TechName= SearchHelp= weitere searchhelp Optionen	Legt eine Suchhilfe für die Spalte fest. Die entsprechenden Optionen sind wie in InputField.

Beispiel Drucktasten (VA01)

```
Column "VH" size=4 position=6 -pushbutton
label="@E8\QVerkaufshilfsmittel@" process="va01_vh.txt"
```

Hierdurch wird eine neue Tabellenspalte mit Drucktasten definiert. Angezeigt wird die Ikone mit id "E8", als Tooltip erscheint "Verkaufshilfsmittel". Bei Klick auf den Button wird das InputScript "va01_vh.txt" aufgerufent. Die Nummer der aktuellen Zeile in der Anzeige (nicht absolut in der gesamten Tabelle) steht in V[_tabrow]. Das InputScript markiert die Zeile und springt dann durch den Funktionscode "=PADD" zu "Verkaufshilfsmittel":

```
// InputScript va01_vh.txt
Setcell[Alle Positionen,0,&V[_tabrow]]"X"
Enter"=PADD"
```

7.9. CompareText

Funktion	Mit CompareText können Sie zwei Textvariablen vergleichen.

Beispiele	`CompareText text1="t1" text2="t2"`
	`if Q[ok]`
	`  ...`
	`endif`

	Der Inhalt der Langtextvariablen "t1" und "t2" wird verglichen.
Format	`CompareText text1="t1" text2="t2"` Vergleicht den Inhalt der Langtextvariablen "t1" und "t2". Das Ergebnis können Sie mit "if Q[ok]" (gleiche Texte) abfragen.

7.10. ConnectHtml

Funktion

Mit der `connectHTML`-Anweisung können Sie Werte aus einer mit `View` angezeigten HTML-Seite lesen, Werte dort verändern und Aktionen in der HTML-Seite auslösen. Sie können dabei eigene HTML-Seiten verwenden oder auch beliebige Internet oder Intranet-Dateien. Die Anzeige über `View` kann entweder innerhalb des SAP Screens erfolgen oder in einem separaten Fenster. Weitere Details finden Sie in der Dokumentation "Special Topics" unter "Automatisierung von Internet-Seiten".

Beispiel

```
connectHTML click="button_Weiter"
```

Der Button "Weiter" in der über `View` angezeigten Internetseite wird gedrückt.

Format

Meist ist nur eine einzige HTML-Seite durch `View` geöffnet. In diesem Fall kann

- `connectHTML optionen`

ohne Angabe eines Fenster `window="..."` genutzt werden. Alle Optionen sowie die Variablen html[...] beziehen sich dann automatisch hierauf. Falls mehr als ein `View` aktiv ist, kann durch die Angabe

- `connectHTML optionen window="&V[viewwindow]"`

explizit ein Fenster genannt werden, das durch "`View .... returnwindow=viewwindow`" davor erhalten wurde. Alle weiteren Optionen sowie die HTML-Variablen html[...] beziehen sich dann auf dieses Fenster.

Eine `connectHTML` Anweisung ist nur nötig, wenn man bestimmte Sonderoptionen (siehe unten) nutzen möchte, oder wenn man das Viewfenster explizit auswählen will. Ansonsten können die HTML-Variablen html[...] auch ohne vorhergehendes `connectHTML` sofort verwendet werden, wenn ein HTML-View aktiv ist. Insbesondere:

- `Set html[name] "wert"` `// Setzt den Wert in das HTML-Element`
- `"... &html[name] ..."` `// liest den Wert aus der HTML-Seite`
- `if html[name]` `// fragt die Existenz des HTMl-Elementes ab`

Optionen

`window=`	Expliziter Bezug auf ein Fenster, das durch `View .... returnwindow=` erhalten wurde.

	Bitte beachten Sie dabei folgendes: Falls Sie in einem GuiXT-Script einen HTML-View in den SAP-Screen neu einblenden und dann unmittelbar im gleichen Script durch `connectHTML` oder über html-Variablen ansprechen wollen, ist die Angabe `returnwindow=` bei `View` erforderlich. Nur dann wird der HTML-View synchron während der Verarbeitung des Scriptes aufgebaut. Ohne `returnwindow=` wäre die Seite erst nach der Anzeige des SAP-Screens ansprechbar. Bei HTML-Views, die Sie in einem separaten Fenster anzeigen durch `View .... -floating` ist das nicht nötig, da diese immer synchron während der Scriptverarbeitung aufgebaut werden.
`click=`	Angabe eines HTML Elements (z.B. ein Button, ein Image oder ein Link) auf den geklickt werden soll. Beispiel: `connectHTML click="image_search.jpg"`

`geturl=`	Die aktuelle URL wird in die angegebene Variable zurückgeliefert. Das ist insbesondere dann nützlich, wenn der Benutzer bereits auf andere Seiten navigiert haben kann. Beispiel:
	`connectHTML geturl="url"`
	`if not Q[ok] or not V[url=http:// ... /basket.asp]`
	`  Return "E: Bitte zunächst den 'Warenkorb' anwählen" -statusline`
	`endif`

`seturl=`
 	
`gettitle=`
	Die HTML-Seite mit der angegebenen URL wird nun angezeigt.
	
	Der Titel der Seite wird zurückgeliefert.

`settitle=`	Der Titel der Seite wird neu gesetzt. Da er bei `View` nicht angezeigt wird, ist das nur zur internen Verwendung in speziellen Fällen sinnvoll.

`listelements=`	Angabe einer Textvariable, in die alle HTML-Elemente der laufenden Seite mit ihren Werten übertragen werden. Das ist insbesondere während der Scriptentwicklung nützlich, um die Elementnamen bequem über Cut&Paste entnehmen zu können. Beispiel:
	`// GuiXT Script`
	`Pushbutton (toolbar) "HTML Elemente lesen" process="get_html_elements.txt"`
	`Textbox (10,0) (40,120) name="elm"`
	
	`// InputScript "get_html_elements.txt"`
	`connectHTML listelements="elm"`

`-shownames`	Sinnvoll während der Entwicklung: In dem HTML-View werden die Namen aller Elemente als Tooltip angezeigt, sobald der Mauszeiger einige Zeit auf dem Element verweilt.

searchbrowserwindow=	Liefert ein offenes Browser Fenster (nur Internet Explorer), oder 0 falls kein Fenster gefunden wurde. Mit
	`if Q[ok]`
	`...`
	`endif`
	können Sie abfragen, ob ein Fenster gefunden wurde. Das Browserfenster kann in allen `connecthtml` Anweisungen ebenso wie ein von `View` zurückgeliefertes Fenster verwendet werden.

index=	In Verbindung mit `searchbrowserwindow=`. Sie geben eine Nummer i an (i=1,2,3,..) zur Suche des Browser-Fensters, falls mehrere existieren.

Ansprechen der HTML-Elemente

Die HTML-Elemente können analog zu den Scriptvariablen V[...] verwendet werden; die Notation ist html[...]. Unterstützt sind:

- `Set html[name] "wert"` `// Setzt den Wert in das HTML-Element`
- `"... &html[name] ..."` `// liest den Wert aus der HTML-Seite`
- `if html[name]` `// fragt die Existenz des HTMl-Elementes ab`

Nach dem Setzen eines Wertes wird automatisch eine eventuell vorhandene "on change" Routine der HTML-Seite aufgerufen. Bei Auswahlfeldern (Select) kann es nötig sein, nach Setzen des Wertes noch das Element durch `click=` anzuklicken, da eventuelle interne Verarbeitungen erst dadurch ausgelöst werden.

Generell ist bei dem automatischen Ablauf von HTML-Seiten zu beachten, dass in der Regel die vorhandenen Funktionen (Setzen von Werten und Klick auf Buttons, Links und Images) ausreichen, die Benutzeraktion zu simulieren. In speziellen Fällen, bei denen über JavaScript, VBScript oder durch Einbinden von Plugins eine über HTML gehende Funktionalität verwendet wird, können diese Möglichkeiten eventuell an ihr Limit stossen.

Namen der HTML-Elemente

Während der Entwicklung empfiehlt sich die Verwendung der Funktionen `listelements=` und `-shownames` zur Anzeige der intern vergebenen Namen.

Generell setzt GuiXT den Namen zusammen aus dem Typ des Elements (image_, button_, checkbox_, ...) sowie einem in der HTML-Seite vergebenen Namen oder dem Text. Eventuelle Leerzeichen werden im Namen durch den Unterstrich ersetzt. Bei duplikativen Namen hängt GuiXT automatisch einen Index .2, .3,... an den ursprünglichen Namen. Beispiele:

```
html[image_help.jpg]
html[link_Suche]
html[link_Suche.2]
html[select_Country]
html[text_Search_Filter]
```

HTMl –

Häufig werden Tabellen in HTML-Seiten genutzt, die auch ineinander geschachtelt werden

Tabellen

können. Die Inhalte aller Tabellenzellen stehen als HTML-Variablen zur Verfügung. Dabei verwendet GuiXT folgende Logik: cell_i.j.k ist der Name der Zelle Nr. k in Zeile Nr. j der Tabelle Nr. i.

Zum Beispiel bezeichnet

```
html[cell_1.4.5]
```

eine Zelle in der ersten Tabelle der Seite, und zwar die Zelle in Reihe 4 Spalte 5.

Bei geschachtelten Tabellen wird pro Zelle das gleiche Verfahren angewandt und der Zellenname davorgesetzt, mit '~' verbunden. Beispielsweise ist

```
html[cell_3_1_7~1_2_2~2.2.4]
```

folgende Zelle:

Die dritte äussere Tabelle auf der Seite. Dort die Zeile 1, Spalte 7. Diese enthält mindestens eine weitere Tabelle. Wir nehmen die erste und gehen in ihr in Zeile 2, Spalte 2. In dieser Zelle sind mindestens zwei weitere Tabellen. Wir nehmen die zweite; dort ist dann in Zeile 2, Spalte 4 die gewünschte Zelle.

Anzahl der Zeilen pro Tabelle

Pro Tabelle enthält eine Variable "..._rows" die Anzahl der Tabellenzeilen. Beispielsweise ist table_3_rows

```
"&html[table_1_rows]"
```

die Anzahl der Zeilen der ersten Tabelle. Bei geschachtelten Tabellen entsprechend. Zum Beispiel ist.

```
"&html[table_3_1_7~1_rows]"
```

die Anzahl der Zeilen der ersten Tabelle in Zelle 1_7 der dritten Tabelle der Seite.

Bei Schleifenkonstrukten in einem Script ist es häufig einfacher, statt der Zeilenzahl die Existenz einer Zelle abzufragen. Beispiel:

```
// Zeilenzähler
Set V[i] 1

label anf

// weitere Zeile vorhanden? sonst Ende der Tabelle
if not html[cell_1.&V[i].1]
  goto end
endif

... Verarbeiten der Tabellenzeile

Set V[i] &V[i] + 1
goto anf

label end
```

Checkbox Die Werte einer Checkbox oder eines Radiobutton werden durch die Werte 0 und 1

und Radiobutton	repräsentiert. Das Setzen eines Radiobutton (Wert 1) löscht automatisch die übrigen Radiobutton der gleichen Gruppe. Beispiel: `Set html[radio_English] 1`

Ladezeiten der HTML-Seite	Nach Aktionen, z.B. einem Klick auf einen Button, wird eventuell eine neue HTML-Seite geladen, was einige Zeit dauern kann. Jede Anweisung
	`connectHTML`
	
	wartet zunächst automatisch, bis die Seite ganz geladen ist, ehe die gewünschte Aktion stattfinden. Nach einer Wartezeit von maximal 2 Minuten beendet `connectHTML` die Warteschleife. Durch
	
	`if not Q[ok]`
	`  Return "...."`
	`endif`
	
	können Sie nach `connectHTML` abfragen, ob die Aktion erfolgreich war.

7.11. Converttext

Funktion	Mit `ConvertText` können Sie eine Langtextvariable von einer Codepage in eine andere konvertieren (ANSI oder UTF-8)
Beispiel	`ConvertText "tx" fromCodepage="ANSI" toCodepage="UTF-8"` Der Inhalt der Langtextvariablen "tx" wird von ANSI nach UTF-8 konvertiert.
Format	`ConvertText "textvar" fromCodepage="cp1" toCodepage="cp1"` Konvertiert den Inhalt der Langtextvariablen "textvar" von Codepage "cp1" nach Codepage "cp2". Folgende Codepage-Angaben sind unterstützt: • ANSI • UTF-8 Sie können entweder `fromCodepage=` oder `toCodepage=` weglassen. GuiXT benutzt für den weggelassenen Parameter dann die Frontend Codepage, d.h. ANSI für nicht-Unicode SAP-Systeme und UTF-8 für Unicode Systeme.

Tipps & Tricks	Um den Inhalt einer Variablen V[myvar] zu konvertieren, können Sie einen Langtext benutzen:
	
	`Set text[x]"&V[myvar]"`
	`ConvertText "x" fromCodepage="ANSI"toCodepage="UTF-8"`
	`Set V[myvar] "&text[x]"`

7.12. CopyFile

Funktion	Mit der `CopyFile`-Anweisung können Sie eine Datei kopieren.
Beispiel	`CopyFile source="C:\images\p100.gif"  destination="C:\temp\p100.gif"` Die Datei `"C:\images\p100.gif"` wird unter Namen `"C:\temp\p100.gif"` kopiert.

Format	`CopyFile source="sourcefile"  destination="destinationfile"`
	Die Source-Datei wird unter dem neuen Namen kopiert. Eine eventuell schon existierende Datei wird dabei überschrieben. Wenn die Source-Datei nicht vorhanden ist, wird keine Aktion durchgeführt.
	Durch
	`if Q[ok]`
	`...`
	`endif`
	können Sie abfragen, ob die Operation erfolgreich war.
	Unter `source=` sind möglich:
	• lokale Dateien, z.B. "C:\temp\abc.txt"
	• Dateien in UNC Notation, z.B. "\\P0012\GuiXT\abc.txt"
	• Dateien auf http oder ftp Servern, z.B. "http://www.synactive.com/demo/logon.gif"
	• Dateien aus dem SAP Web Repository, z.B. "SAPWR:ZGUIXT.logon.gif"
	Und unter `destination=` können Sie angeben:
	• lokale Dateien, z.B. "C:\temp\abc.txt"
	• Dateien in UNC Notation, z.B. "\\P0012\GuiXT\abc.txt"
	• Dateien auf http oder ftp Servern, z.B. "ftp://synactive.com/demo/logon.gif"
	• Aber keine Dateien aus dem SAP Web Repository
	Benutzer und Passwort können bei http/ftp wie folgt mitgegeben werden:
	• "ftp://user:password@synactive.com/demo/logon.gif"

7.13. CopyText

Funktion	Mit der `CopyText`-Anweisung können Sie Texte zwischen Dateien, Bildschirmbereichen und Textvariablen kopieren.
Beispiele	`CopyText fromText="t1" toScreen="X[LONGTEXT]"` Der Text wird aus der Textvariable `"t1"` in den Textbereich `X[LONGTEXT]` auf dem Bildschirm

kopiert.

Format

```
CopyText from...="name1" to...="name2"
```

Möglich sind: `fromText, fromScreen, fromFile, fromTemplate, fromString, -fromCLipboard, fromSharedText, fromDirectory, toText, toScreen, toFile, toString, -toClipboard, toSharedText.`

Dabei muss entweder `fromText` oder `toText` (oder beides) angegeben sein. Bedeutung:

`fromText="t1"`	`toText="t2"`	Der Text `t1` wird in den Text `t2` kopiert
`fromText="t1"`	`toScreen="X[area]"`	Der Text `t1` wird in den Bildschirmbereich `X[Area]` kopiert. **Bitte beachten**: Erst bei Anzeige des Bildschirms bzw. bei "`Enter`" in einem InputScript wird der Text tatsächlich kopiert (asynchrone Verarbeitung)
`fromText="t1"`	`toFile="dsn"`	Der Text `t1` wird in die Datei `dsn` kopiert
`fromText="t1"`	`toString="varname"`	Der Beginn des Textes Textes `t1` wird in die Variable `varname` kopiert
`fromText="t1"`	`-toClipboard`	Der Text wird in das Windows clipboard (Zwischenablage) kopiert
`fromText="t1"`	`toSharedText="s1"`	Der Text `t1` wird in den SAP-Modus-unabhängigen Text `s1` kopiert
`toText="t1"`	`fromSharedText="s1"`	Der SAP-Modus-unabhängige Text `s1` wird in den Text `t1` kopiert
`toText="t1"`	`fromScreen="X[area]"`	Der Langtext in Bildschirmbereich `X[Area]` wird in den Text kopiert. **Bitte beachten**: Erst bei Anzeige des Bildschirms bzw. bei "`Enter`" in einem InputScript wird der Text tatsächlich kopiert (asynchrone Verarbeitung)
`toText="t1"`	`fromFile="dsn"`	Die Datei `dsn` wird in den Text `t1` kopiert. Es kann sich dabei auch um Dateien auf http oder ftp-Servern oder aus dem Web Repository handeln (Notation: http://..., ftp://..., SAPWR:...).
`toText="t1"`	`fromDirectory="dsn"`	Eine Liste aller Dateien in Verzeichnis `dsn` wird in den Text `t1` kopiert. Dabei können die

			Platzhalter * und ? angegeben werden, z.B. fromDirectory="C:\GuiXT*.gif", analog dem MS DOS "dir" Kommando. Unterverzeichnisse werden ignoriert, siehe aber Option "-subdirectories"
	toText="t1"	fromTemplate="dsn"	Die Datei dsn wird in den Text t1 kopiert; dabei werden alle &-Variablen durch die entsprechenden Inhalte ersetzt.
	toText="t1"	fromString="varname"	Der Inhalt der Variable varname wird in den Text kopiert
	toText="t1"	-fromClipboard	Der Inhalt des Windows clipboard (Zwischenablage) wird in den Text kopiert

Optionen	-append	Der Text wird an einen bestehenden Text angehängt.
	-appendline	Der Text wird als neue Zeile an einen bestehenden Text.
	line=	Nur in Verbindung mit toString=möglich. Angabe einer Zeilennummer, beginnend bei 1, die aus dem Text ausgelesen werden soll. Durch if Q[ok] kann nach CopyText abgefragt werden, ob die angeforderte Zeile gefunden wurden.
	delimiter=	Nur in Verbindung mit toString= und line= möglich. Statt des Zeilenvorschubs wird das angegebene Zeichen als Begrenzung betrachtet.
	-subdirectories	Nur in Verbindung mit fromDirectory=. Statt der Dateinamen werden alle Unterverzeichnisse bereitgestellt.
Tipps & Tricks	Eine Reihe von Beispielen finden Sie in dem Artikel "Texterfassung und InputScripts" in den "Special Topics".	

7.14. Decimalseparator

Funktion	Mit der Decimalseparator-Anweisung können Sie das in Rechnungen mit Set benutzte Zeichen zur Abtrennung der Kommastellen festlegen ('.' oder ','). Als Default nimmt GuiXT bei Anmeldesprache Englisch den Punkt, ansonsten das Kommazeichen an.
Beispiel	Decimalseparator "."
Format	Decimalseparator "." Decimalseparator ","

	`Decimalseparator UserDefault` Der im Benutzerstammsatz abgelegte Default (Punkt oder Komma) wird benutzt. Beim ersten Verwenden dieser Anweisung wird der Defaultwert über einen Remote Function Call (RFC) eingelesen. Es ist daher ein gültiger RFC Benutzer im GuiXT-Profile erforderlich.

7.15. DropDownList

Funktion	Mit `DropDownList` können Sie die Eingabe in einem SAP ERP Standardfeld oder für eine eigene Variable als Auswahlliste anbieten.

Beispiele	`DropDownList (10,1) "currencylist" refer="F[Währung]" width=20`
	
	Es wird eine Auswahlliste mit Werten aus dem Langtext "`currencylist`" angezeigt. Der aktuelle Wert wird dem Feld F[Währung] entnommen und bei Selektion eines neuen Wertes dort zurückgestellt.
	
	`DropDownList (4,1) "countrylist" refer="V[XD_country]" width=24`
	
	Es wird eine Auswahlliste mit Werten aus dem Langtext "`countrylist`" angezeigt. Der aktuelle Wert wird der Variablen V[XD_country] entnommen und bei Selektion eines neuen Wertes dort zurückgestellt.

Format	`DropDownList (Zeile,Spalte) "longtext" refer="name"` Die Auswahlliste wird wie folgt aus dem Langtext "longtext" aufgebaut: • Jede Textzeile bildet einen Wert der Auswahlliste • Optional können Sie auch mehrere Werte in einer Textzeile unterbringen; dann bitte jeweils durch ein Semikolon voneinander trennen • Es kann entweder nur der Wert abgelegt sein, z.B. ein Währungsschlüssel EUR, SFR, ... Oder Sie legen zusätzlich einen beschreibenden Text ab. Nur der Text ist dann in der Auswahlliste sichtbar, während der Wert als Inhalt des Feldes oder der Variable dient. Das Format ist: wert=text, z.B. "BV=Barverkauf". • Sie können auch für einen leeren Wert einen Text aufnehmen. Er wird angezeigt, wenn das Feld bzw. die Variable leer sind. Beispiel: "=--- Auswahl Auftragsart ---" Den Langtext können Sie im Script mit festen Werten füllen (siehe Beispiel 1), aus einer Datei einlesen, in der Sie die Werte separat pflegen, oder aus der SAP ERP Datenbank lesen, z.B. als Werte einer SAP Suchhilfe (Beispiel 2).
Optionen	`process=` Angabe eines InputScripts, das bei Änderung der Auswahl in der dropdown Liste aufgerufen wird. `width=` Breite der Auswahlliste (Anzahl Spalten). Default ist 16.

Textfont=	Gewünschter Font. Default ist "Arial" .
Textheight=	Texthöhe in Pixel. Der Default ist abhängig von der eingestellten SAP GUI Fontgrösse.
Textweight=	Schriftdicke zwischen 1 (dünn) bis 9 (sehr dick). Default ist 5.
-readonly	Nur der aktuelle Eintrag wird angezeigt, keine Änderung möglich

Beispiel 1

```
// VA01 Auftragsart
del F[Auftragsart]-triple
Set text[ordertypes] "=--- Auswahl Auftragsart ---
;TA=Terminauftrag;BV=Barverkauf;RE=Retoure;"
DropDownList (2,0) "ordertypes" refer="F[Auftragsart]" width=30
```

Beispiel 2

```
// Auswahl Länderschlüssel
if not text[ddl_country]
include "generate_ddl_country.txt"
endif

DropDownList (6,1) "ddl_country" refer=V[XD_country] width=30
```

In dem include "generate_ddl_country.txt" generieren wir die Liste der Länderschlüssel und Texte aus der SAP ERP Datenbank. Wir nutzen dazu die Funktion "Z_S10_SEARCHHELP", die Synactive im Rahmen des Synactive S10 Systems (http://www.synactives10.com/s10forum) geschrieben hat. Sie liest die Wertehilfe aus dem SAP System und kann sehr einfach in GuiXT Skripten benutzt werden.

```
// include file "generate_ddl_country.txt": Generieren Länderschlüssel aus
SAP Suchhilfe
Set text[ddl_country] "=--- Select a country ---"

Call "Z_S10_SEARCHHELP" in.SEARCHHELP="H_T005"
in.COLUMNS="LAND1(8),LANDX(30)" table.DATA="data"

Set V[i] 1
label generate_country_list_next
CopyText fromText="data" toString="line" line=&V[i]
if not Q[ok]
goto generate_country_list_done
endif

Set V[item] "&V[line](1,8)=&V[line](9,38)"
CopyText fromString="item" toText="ddl_country" -appendline

Set V[i] &V[i] + 1
goto generate_country_list_next

label generate_country_list_done
```

7.16. Enter

Funktion	Mit der **Enter**-Anweisung wird innerhalb eines InputScript die weitere Verarbeitung gesteuert. **Enter** ohne weitere Zusätze simuliert die Enter-Taste. Optional können Sie einen internen Funktionscode mitgeben; dieser entspricht der Auswahl eines Menüeintrags oder dem Drücken einer Drucktaste.	
Beispiele	**Enter** **Enter "BU"** Der interne Funktionscode **"BU"** wird ausgelöst. **Enter "/nSE38" process="ShowABAPSource.txt"** Die Transaktion **SE38** wird aufgerufen und das InputScript **ShowABAPSource.txt** wird ausgeführt	
Format	**Enter** **Enter "Wert"** **Enter "Wert" process="InputScript.txt"**	
Optionen	**OnError=**	Angabe eines Funktionscodes, der dann ausgeführt wird, wenn der erste angegebene Funktionscode zu einer Fehlermeldung geführt hat. Ein Beispiel hierzu finden Sie in Special Topics->Die Behandlung von Dateien in InputScripts, Example 2. Der unter **OnError=** angegebene Funktionscode wird häufig "/N" (Abbruch der Transaktion) oder "/12" (Abbrechen) sein. Das Script wird mit der entsprechenden Screen-Anweisung, nach Ausführen des OnError-Funktionscodes, fortgesetzt. Die Variable **V[_lasterror]** enthält dann den Fehlertext. **OnError="Continue"** Das ist die flexibelste Möglichkeit der Fehlerbehandlung, es müssen jedoch alle folgenden Aktionen in dem InputScript selbst programmiert werden. Nach einer Fehlermeldung wird das InputScript nach der **Enter**-Anweisung fortgesetzt. Die Variable **V[_lasterror]** enthält dann den Fehlertext. Sie können beispielsweise nun Werte durch "Set" korrigieren und mit "Enter" einen neuen Funktionscode auslösen. Bitte beachten Sie: • Nicht alle normalen Eingabefelder sind auch nach der Fehlermeldung noch eingabebereit • Alle Anweisungen zwischen **Enter ... OnError="Continue"** und der nächsten **Screen** Anweisung werden nur ausgeführt, falls **Enter** zu einer Fehlermeldung geführt hat, ansonsten übersprungen. • Sie können **Enter ... OnError="Continue"** auch

		mehrfach in einem `Screen` Block verwenden, z.B. um mehrere Eingabewerte zu versuchen. Beispiel:
		`Screen ...`
		`  Set F[Material] "abc"`
		`  Enter "/5" OnError="Continue"`
		`  // Die folgenden Anweisungen werden nur nach einer Fehlermeldung ausgeführt`
		`  Message "Material abc not found"`
		`  Set F[Material] "xyz"`
		`  Enter "/5" OnError="Continue"`
		`  // Die folgenden Anweisungen werden nur nach einer zweiten Fehlermeldung ausgeführt`
		`  Message "Material xyz not found"`
		`  Set F[Material] "M-01"`
		`  Enter "/5"`

	`Wait=`	Parameter in Millisekunden zwischen 0 und 5000. Die Enter-Taste wird erst nach der angegebenen Zeit gedrückt.
	`Enter "/ScrollToLine=..."` `Table=`	Angabe einer Tabelle auf dem Bildschirm sowie einer Zeilennummer (beginnend bei 1), zu der geblättert werden soll. In Listausgaben kann ohne Angabe einer Tabelle positioniert werden.
	`Enter control="controlname" item="itemid" event="eventno"`	Löst in speziellen Controls (z.B. "Objektdienste") einen Button oder die Auswahl eines Menüeintrags aus. Bitte verwenden Sie die "Input Recording" Funktion zum Aufzeichnen der korrekten Angaben (control, item und event). Beispiel: `Enter control="DIALOGCTRL.Toolbar" item="CREATE_ATTA" event="2"` Damit ist für relativ einfache Aktionen eine Automatisierung spezieller SAP-Controls direkt im InputScript möglich. Komplexere Abläufe erfordern den Aufruf des "SAP GUI Scripting"; siehe dazu den Artikel in den "Special Topics".
Tips &Tricks		• Falls in einem InputScript für ein Bild kein `Enter` angegeben ist, wird an dieser Stelle das InputScript beendet und der Benutzer kann manuell die Transaktion fortführen. • Wie Sie den internen Code erhalten, ist bei `Pushbutton` näher beschrieben • Falls Sie zum Test, durch Einschalten von **Options->Visible Auto Processing** die Transaktion mit dem InputScript sichtbar ablaufen lassen, wird durch Drücken der Enter-Taste ebenfalls der von Ihnen gesetzte Funktionscode ausgelöst.

- Achtung: Wenn `Enter` immer durchlaufen wird, loopt das System. In diesem Fall kann man durch Einschalten von **Options->Visible Auto Processing** beenden.

7.17. GetFieldAttribute

Funktion

Mit der `GetFieldAttribute`-Anweisung können Sie Eigenschaften eines Ein/Ausgabefeldes oder einer Tabellenspalte abrufen.

Beispiele

```
GetFieldAttribute F[Business area.] row="frow"
```

In die Variable `V[frow]` wird der Nummer der Zeile, in der sich das Feld befindet, gestellt.

```
GetFieldAttribute [Alle Positionen,12] header="chead"
```

In die Variable `V[chead]` wird die Spaltenüberschrift der zwölften Spalte der Tabelle "Alle Positionen" gestellt.

Format

```
GetFieldAttribute F[element]options
GetFieldAttribute [column] options
```

Es können eine oder mehrere der unten angegebenen Optionen angegeben werden.

```
GetFieldAttribute cell[table,column,row] options
```

Optionen

`row=`	Zeilennummer
`column=`	Spaltennummer
`textrow=`	Zeilennummer des Feldtextes (meist identisch mit row=)
`textcolumn=`	Spaltennummer des Feldtextes
`text=`	Feldtext
`size=`	Feldlänge
`techname=`	Technischer Name z.B. VBKD-BSTKD
`header=`	Bei Spalten: Spaltenüberschrift
`columnnumber=`	Bei Spalten: Spaltennummer in Tabelle
`displaycolumnnumber=`	Bei Spalten: Spaltennummer in Anzeige
`width=`	Bei Spalten und speziellen Controls: Breite
`height=`	Bei speziellen Controls: Höhe
`buttontext=`	Bei Drucktasten: Text auf dem Button
`buttonicon=`	Bei Drucktasten: Id der Ikone
`buttontooltip=`	Bei Drucktasten: Tooltip
`protected=`	Bei Ein/Ausgabefeldern: 'X' = nicht eingabebereit. Bei Drucktasten: 'X' = nicht aktiv.

7.18. GetTableAttribute

Funktion	Mit der `GetTableAttribute`-Anweisung können Sie verschiedene Attribute einer Bildschirmbildtabelle lesen.
Beispiel	`GetTableAttribute T[Table] FirstVisibleRow=FVisRow LastVisibleRow=LVisRow LastRow=LastRow` Die gewünschten Attribute der Tabelle werden in die angegebenen Variablen `V[FVisRow]`,, `V[LVisRow]`, `V[LastRow]` gestellt.
Format	`GetTableAttribute tabid ... attribut=varname ...` Folgende Attribute können gelesen werden: `FirstVisibleRow=` Erste auf dem laufenden Bildschirmbild angezeigte Zeile, bei 1 beginnend `LastVisibleRow=` Letzte auf dem laufenden Bildschirmbild angezeigte Zeile `LastRow=` Insgesamt letzte Zeile der Tabelle (unabhängig von der aktuellen Anzeige) Beispielsweise ist zu Beginn der Anzeige FirstVisibleRow=1, LastVisibleRow=12, LastRow=115, d.h. die Zeilen 1 bis 12 einer Tabelle werden angezeigt und die Tabelle hat insgesamt 115 Zeilen.
Tips &Tricks	• Ein ausführliches Beispiel zum Scrollen durch eine Tabelle finden Sie unter "Tips&Tricks (9)".

7.19. Goto/Label

Funktion	Mit der `Goto`-Anweisung können Sie die Verarbeitung innerhalb eines InputScript steuern.

Beispiel	`Goto NextPosition`
	`Label NextPosition`
	`Screen SAPMVCF66G.0400`

Format	`Goto labelname`
	`Label labelname`
	
	Die Verarbeitung des InputScript wird mit der auf das angegebene Label folgenden Zeile fortgesetzt.

Tips &Tricks	• Rückwärtssprünge mit `Goto` sind möglich. Sie sind gefährlich, da Sie sehr leicht eine Endlosverarbeitung damit produzieren. Falls Sie beim Test eine solche Situation haben, können Sie durch Einschalten der Option Visible auto processing in GuiXT die Verarbeitung stoppen und mit einem /N-Kommando neu starten.

	• Falls die **Label**-Anweisung vor einer **Screen**-Anweisung steht, wird die Verarbeitung des aktuellen Bildschirms nach Ausführung des **Goto** beendet. Steht sie dagegen hinter einer **Screen**-Anweisung, so wird die Verarbeitung auch dann fortgesetzt, wenn die Screen-Namen und -Nummern unterschiedlich sind. • Es ist möglich, **if/else/endif** innerhalb eines durch **Screen** eingeleiteten Anweisungsblocks zu verwenden. Dagegen ist nicht unterstützt, innerhalb von **if/else/endif** eine **Screen**-Anweisung anzugeben. Stattdessen muß für die Navigation zwischen Screens **Goto** eingesetzt werden.

7.20. HistoryUpdate

Funktion	Mit der **HistoryUpdate**-Anweisung können Sie die Aufzeichnung der Eingabehistorie abschalten oder wieder einschalten, entweder für alle Eingabefelder oder für ein spezielles Feld, z.B. ein Datumsfeld.
Beispiele	**HistoryUpdate F[Buchungsdatum] off**
Format	**HistoryUpdate off / on** **HistoryUpdate F[Feld] off / on** Schaltet die Aufzeichnung der Eingabehistorie für den ganzen Bildschirm oder für ein einzelnes Feld ab oder wieder an.

7.21. Inputfield

Funktion	Mit der **InputField**-Anweisung können Sie eigene Eingabefelder definieren. Der dort eingetragene Wert kann in einem InputScript verwendet werden.	
Beispiel	**InputField (10,1)"Referenznummer" (10,20) Size="10" Name="Refnr"** Hierdurch wird ein 10-stelliges Eingabefeld definiert. Der Feldtext wird an Position (10,1) angezeigt, das Eingabefeld selbst an Position (10,20). Der interne Feldname ist Refnr, d.h. in einem InputScript wird das Feld in der Form &[Refnr] genannt.	
Format	**InputField (Zeile1,Spalte1) "Feldtext" (Zeile2,Spalte2) Size="..." Name="..."**	
Zusatzoptionen	**-Numerical**	Nur Zifferneingabe möglich
	-Uppercase	Eingabe nur in Grossbuchstaben
	-date	Nur Datumseingabe möglich.
	-NoLabel	Eingabefeld ohne Text. Die Angaben **(Zeile1,Spalte1)** und

	"Feldtext" entfallen dann.
-dynamicLabel	In Verbindung mit TechName=: Der angegebene Text wird durch den im SAP Repository abgelegten Feldtext überschrieben. Je nach verfügbarem Platz wird der kurze, mittlere oder lange Text gewählt. Im GuiXT Profile muss dazu ein RFC Benutzer mit entsprechenden Berechtigungen hinterlegt sein.
-Invisible	Unsichtbare Eingabe (Passwort)
-Required	Das Feld wird als Mussfeld (obligatorische Eingabe) dargestellt. Eine automatische Prüfung auf Eingabe findet nicht statt, sie muss selbst im InputScript durchgeführt werden.
-Intensified	Der Wert wird farblich hervorgehoben (Standard: rot)
-AlignRight	Der Wert wird rechtsbündig ausgegeben
-LeadingSpace	Führende Leerzeichen werden übernommen
-ReadOnly	Nur Ausgabe
Default=	Angabe eines Defaultwertes
MaxLength=	Maximale Eingabelänge. Hier kann eine größere Länge als in Size= angegeben werden; das Feld wird dann automatisch scrollbar.
TechName= SearchHelp= -nosearchhelp shname=	Angabe eines Feldnamens aus dem SAP Repository, z.B. TechName="BSEG-GSBER". Es wird dann bei F4 bzw. Matchcode-Anwahl der SAP-Standard-Matchcode prozessiert und bei F1 die Standard-Feldhilfe. Alternativ kann auch direkt der Name einer Suchhilfe aus SAP durch SearchHelp="..." angegeben werden. Falls durch techname= nur die F1-Hilfe, aber keine F4-Suchhilfe zugeordnet werden soll, können Sie die Option -nosearchhelp verwenden. Falls die Suchhilfe mehr als ein Feld liefert, nimmt GuiXT das erste Feld. Optional kann durch shname= der gewünschte Feldname aus der Suchhilfe direkt angegeben werden, z.B. shname="MATNR" **Die Installation des ABAP Programms ZGUIXTF4 ist hierzu**

	erforderlich (verfügbar auf Download-Seite)

shselname1=
shselvalue1=
 	
shselname2=
shselvalue2=
 	
shselname3=
shselvalue3=
	Angabe von bis zu 3 Selektionskriterien, die bei der Anzeige der Matchcodeliste berücksichtigt werden. Die unter **shselname1, shselname2, shselname3** angegebenen Namen müssen so wie in der Suchhilfe (Transaktion SE11) definiert angegeben werden. Als Werte **shselvalue1, shselvalue2, shselvalue3** können entweder Direktwerte oder andere Eingabefelder [...] angegeben werden. Beispiele:
	
	`InputField .... searchhelp="COCAN" shselname1="GJAHR" shselvalue1="2002"`
	
	Es werden nur Kostenstellen mit Jahr "2002" zur Auswahl angeboten
	
	`InputField .... searchhelp="COCAN" shselname1="GJAHR" shselvalue1="F[Jahr]"`
	
	Der aktuelle Inhalt des Eingabefeldes **F[Jahr]** wird zur Selektion des Jahres im Matchcode gesetzt.
	
	`InputField .... searchhelp="COCAN" shselname1="GJAHR" shselvalue1="V[Varname]"`
	
	Der aktuelle Inhalt der Variable **V[Varname]** wird zur Selektion des Jahres im Matchcode gesetzt.

shname1=
shdest1=
 	
shname2=
shdest2=
 	
shname2=
shdest2=
	Angabe von bis zu 3 weiteren Feldern, die nach Auswahl eines Matchcode-Eintrages ebenfalls gefüllt werden. Die unter **shname1, shname2, shname3**angegebenen Namen müssen so wie in der Suchhilfe (Transaktion SE11) definiert angegeben werden. Als Zielfeld **shdest1, shdest2, shdest3**können entweder Eingabefelder **F[Feldname]** oder Variablen **V[Varname]** angegeben werden. Beispiele:
	
	`InputField .... searchhelp="COCAN" shname1="LSTAR" shdest1=F[Leistungsart]`
	
	Nach Auswahl der Kostenstelle wird diese in das neu definierte InputField gestellt. Zusätzlich wird die Leistungsart aus dem selektierten Eintrag in das Eingabefeld **Leistungsart** übernommen.
	
	`InputField .... searchhelp="COCAN" shname1="LSTAR" shdest1=F[Leistungsart] shname2="KTEXT_KOSTL" shdest2=V[CC_text]`
	
	Zusätzlich wird die Variable **V[CC_text]** mit dem Kostenstellentext gefüllt, der für die weitere Verarbeitung genutzt werden kann.

	Hinweis: Bitte hierzu das ABAP Programm ZGUIXTF4 in der Version 2002 Q3 6 oder höher einsetzen.
SearchHelpProcess=	Angabe eines InputScripts, das nach Auswahl eines Matchcodeeintrags prozessiert wird. Folgende Anwendungsfälle sind denkbar: • Anzeige eines Langtextes zu dem ausgewählten Wert: Angabe SearchHelpProcess="return.txt", wobei das InputScript return.txt lediglich die Anweisung "Return" enthält. Hierdurch wird der Bildschirm neu angezeigt, und die durch den Matchcode gesetzten Variablen (wie **V[CC_text]** im vorangehenden Abschnitt) werden dem Benutzer angezeigt. • Anzeige weiterer abgeleiteter Daten zu dem ausgewählten Wert: Das angegebene InputScript besorgt (z.B. durch eine Call-Anweisung) weitere Daten und kehrt durch "Return" zur Anzeige zurück. Die neu besorgten Daten können durch das zum Bild gehörende GuiXT-Skript dargestellt werden.

Eigene Matchcodehilfe

Sie können dem InputField auch eine selbst programmierte Matchcodehilfe zuordnen durch die Notation
searchhelp="exit.progname.formname"
wobei **progname** der ABAP Programmname ist und **formname** der Name der Unterroutine (FORM-Routine), die aufgerufen wird. Die Unterroutine wird mit einer standardisierten Schnittstelle (s.u.) aufgerufen und liefert den ausgewählten Wert zurück. Auch die über **shselname1=, shselvalue1=, ..., shname1=, shdest1=** möglichen Zusatzparameter können in der selbst programmierten Matchcodehilfe behandelt werden (siehe Beispiel in den "Special Topics").

Beispiel 1: Lesen von Werten aus SAP Datenbank

```
Inputfield (18,1) "Bk" (18,20) name="buk" size="4"
searchhelp="exit.zguixtmc1.mcbuk"
```

ABAP-Programm:

```
Program ZGUIXTMC1.

* Value table
data: begin of t1 occurs 100,
bukrs like t001-bukrs,
butxt like t001-butxt,
land1 like t001-land1,
ort01 like t001-ort01,
end of t1.

* Returned value
```

```
data: begin of r1 occurs 1.
include structure DDSHRETVAL.
data: end of r1.

Form mcbuk
tables
sel "table with shselname1=, shselvalue1=,
dest "table with shname1=, shdest1
using invalue
changing selvalue.

* Select data
Select bukrs butxt land1 ort01 from t001
into corresponding fields of table t1.

* SAP standard function to display table as matchcode selection
CALL FUNCTION 'F4IF_INT_TABLE_VALUE_REQUEST'
EXPORTING
RETFIELD = 'BUKRS'
WINDOW_TITLE = 'Please select a company code'
VALUE_ORG = 'S'
TABLES
VALUE_TAB = T1
RETURN_TAB = R1
EXCEPTIONS
OTHERS = 1.

* Return selected value
if sy-subrc = 0.
selvalue = r1-fieldval.
endif.

endform.
```

Beispiel 2: Lokaler Dateiname

```
Inputfield (6,1) "Filename" (6,20) name="filename" size="50"
searchhelp="exit.zguixtmc1.mcfile"
```

ABAP-Programm:

```
Program ZGUIXTMC1.

Data: tmp_filename(80).

Form mcfile
tables
sel "table with shselname1=, shselvalue1=,
dest "table with shname1=, shdest1
using invalue
changing selvalue.

Call Function 'WS_FILENAME_GET'
Exporting
DEF_FILENAME = invalue
DEF_PATH = 'C:\GuiXT'
MASK = ',*.*,*.*.'
MODE = 'O'
TITLE = 'Please select a file '
Importing
FILENAME = TMP_FILENAME
Exceptions
others = 1.
```

```
* Return selected value
If sy-subrc = 0.
selvalue = tmp_filename.
Endif.

Endform.
```

Tipps & Tricks

- Sie können bei dem Feldtext eine Quickinfo hinterlegen, indem Sie den Text in der SAP-Konvention für Quickinfos aufbauen, z.B.:

```
"@0L\QBitte die externe Dokumentennummer angeben, falls
bekannt@Referenznummer"
```

- Der InputAssistant verwaltet auch die Eingabehistorie der selbst definierten Felder
- Den für die Matchcode-Funktionalität erforderlichen technischen Namen finden Sie im SAP in der Hilfe für das betreffende Feld (F1) unter "Technische Informationen".
- Folgende speziellen Suchhilfen können durch eine Angabe unter `SearchHelp="..."` ebenfalls aktiviert werden:
  - ... `searchhelp="GuiXT.ORGEH"` Hierarchische Sicht der Organisation
  - ... `searchhelp="GuiXT.ZTERM"` Zahlungsbedingungen
  - ... `searchhelp="GuiXT.ZTERM.D"` Zahlungsbedingungen, nur Debitoren
  - ... `searchhelp="GuiXT.ZTERM.K"` Zahlungsbedingungen, nur Kreditoren
  - ... `searchhelp="GuiXT.DATE"` Datum (Kalender)
  - ... `searchhelp="GuiXT.TIME"` Zeitauswahl
  - ... `searchhelp="GuiXT.AKONT.D" shselname1="BUKRS" shselvalue1="&F[::Buchungskreis]"` Abstimmkonto Debitoren. Der Buchungskreis muss zwingend mitgegeben werden (als Konstante oder Variable)
  - ... `searchhelp="GuiXT.AKONT.K" shselname1="BUKRS" shselvalue1="&F[::Buchungskreis]"` Abstimmkonto Kreditoren. Der Buchungskreis muss zwingend mitgegeben werden (als Konstante oder Variable)
- Sie können die Eingabe eines Datums wie folgt überprüfen (wir setzen hier ein Datum in der Form dd.mm.yyyy voraus):

```
InputField

(5,5) "Datum"(5,20) name=d1 -date
```

```
if
not V[d1]

Message "E: Bitte das Datum eingeben" -statusline
else

Set V[x]&V[d1]-&V[today_d.m.y]

Set V[x]&V[today_d.m.y]+&V[x]

if not V[x=&V[d1]]

Message "E: Bitte ein gültiges Datum im Format dd.mm.yyyy
eingeben, z.B. 12.05.2010" -statusline

endif
endif
```

7.22. Leave

Funktion	Mit der **Leave**-Anweisung können Sie die Interpretation eines InputScriptes beenden. Das aktuelle Bildschirmbild wird angezeigt. Im Unterschied zu **Return** erfolgt keine Rückkehr zu der Starttransaktion.
Beispiel	**Leave**
Format	**Leave**
Tips &Tricks	Es ist meist sinnvoll, die **Leave**-Anweisung im Anschluss an **Enter** zu benutzen. Wenn Sie das InputScript im Anschluss an eine **Screen**-Anweisung ohne **Enter** mit **Leave** abbrechen, wird das aktuelle Bildschirmbild angezeigt, ohne ein eventuell dazugehörendes GuiXT Script zu prozessieren, was meistens nicht gewollt ist.

7.23. Link

Funktion	Mit der **Link**-Anweisung können Sie Variablen mit Eingabefeldern verbinden. Im Unterschied zu **Set** wird der vom Benutzer eingegebene Wert automatisch wieder in die Variable übernommen. Das ist in einem InputScript besonders dann praktisch, wenn man Werte automatisch in Felder füllt und gleichzeitig damit rechnet, daß der Benutzer den Wert auch manuell eingeben kann, etwa nach einem Fehler. Insbesondere wenn SIe Transaktionen vereinfachen und Eingaben aus späteren Bildschirmen durch eigene InputFields realisieren, kann es nötig sein, auf den normalerweise nicht gezeigten Bilderschirmen durch **Link** eine

	Verbindung zwischen dem selbst definierten InputField und dem Original-Eingabefeld herzustellen, falls der Benutzer auf irgendwelchen Wegen auf den Bildschirm mit dem Originalfeld gelangen kann. Es gibt zwei verschiedene Formen, je nachdem, ob bei Anzeige des Bildschirms der Inhalt der Variablen in das Feld gestellt werden soll, oder der Feldinhalt in die Variable übernommen wird.

Beispiel	`Link F[Datum] V[meinDatum]`
	
	Das Eingabefeld "Datum" erhält den Wert der Variablen "meinDatum", identisch zu `Set F[Datum] "&V[meinDatum]"`. Zusätzlich wird nach Eingabe eines neuen Wertes in das Feld "Datum" durch den Benutzer der Wert automatisch auch in die Variable "meinDatum" gestellt.
	
	`Link V[meinDatum] F[Datum]`
	
	Die Variable "meinDatum" erhält den Inhalt des Feldes "Datum", identisch zu `Set V[meinDatum] "&F[Datum]"`. Zusätzlich wird nach Eingabe eines neuen Wertes in das Feld "Datum" durch den Benutzer der Wert automatisch auch in die Variable "myDate" gestellt.

Format	`Link F[Eingabefeld] V[Varname]` // setzt einen Wert für ein Eingabefeld und übernimmt Eingaben in die Variable `Link V[Varname] F[Eingabefeld]` // setzt den Wert des Feldes in die Variable und übernimmt Eingaben in die Variable `Link cell[Tabelle,Spalte,Zeile] V[Varname]` // setzt einen Wert für eine Tabellenzelle und übernimmt Eingaben in die Variable `Link V[Varname] cell[Tabelle,Spalte,Zeile]` // setzt den Wert der Tabellenzelle in die Variable und übernimmt Eingaben in die Variable

Tips&Tricks	Ein Beispiel finden Sie im Special Topic Felder von einem Tabellenreiter zum andern verschieben
	
	Um den Wert eines Feldes vor und nach der Dateneingabe vergleichen zu können, können Sie `Link` in Verbindung mit einer zusätzlichen Variablen verwenden, in der Sie den Wert vor der Dateneingabe speichern. Beispiel Skript:
	
	`// GuiXT Script`
	`Set V[Menge_alt] "&F[Menge]"`
	`Link V[Menge_neu] F[Menge]`
	
	`// InputScript`
	`// Menge verändert?`
	`if not V[Menge_neu=&V[Menge_alt]]`
	`// Aktion bei verändertem Wert`

```
endif
```

7.24. Localvaluehelp

Funktion	Mit der `LocalValueHelp`-Anweisung können Sie einem Feld dynamisch eine lokale Wertehilfe zuordnen. Es kann sich um ein SAP-Standardfeld oder ein eigenes InputField handeln.
Beispiele	`LocalValueHelp F[Dokumentenart] domain="EXT_DOKTYPE"`

Format	`LocalValueHelp F[Eingabefeld] domain="DOMNAME"`
	Die lokale Wertehilfe wird unter dem Dateinamen "dom.D.DOMNAME.txt" in dem unter "Domains" im GuiXT Profile angegebenen Verzeichnis gesucht. Dabei steht "D" für den Sprachenschlüssel der Anmeldesprache (D=Deutsch, E=Englisch,...).
	Bei eigenen Eingabefeldern geben Sie als Feldnamen den Text des Eingabefeldes (Label) an. Beispiel:
	`InputField (10,1)"Referenznummer" (10,20) size="10" name="Refnr"`
	`LocalValueHelp F[Referenznummr] domain="REFERENZNR"`
	Bei Eingabefeldern ohne Text bitte den Variablennamen angeben. Beispiel:
	`InputField (10,1) size="10" name="MM02_Refnr2" -nolabel`
	`LocalValueHelp F[MM02_Refnr2] domain="REFERENZNR"`

7.25. On

Funktion	Mit der `On`-Anweisung können Sie die Ausführung eines InputScripts, nach dem Drücken der Enter- oder einer anderen Funktionstaste durch den Benutzer, auslösen. `On` wird in GuiXT Skripten eingesetzt.
Beispiele	`On "Enter" Process="nextproduct.txt"` `On "/11" Fcode="/0" Process="update.txt"`
Format	`On "fc" Fcode="newfc" Process="script"` `"fc"` kann einer der folgenden Strings sein: • `"Enter"` • `"/n"` n ist die Nummer der Funktionstaste • `"fcode"` "fcode" ist der durch einen Pushbutton ausgelöste Funktionscode • `"/Menu=i,j,k,l"` i,j,k,l stellen die Menüeingaben dar. Bitte verwenden Sie die Option "Input Recording" um die richtige Menüfolge zu generieren. `Fcode="newfc"` der Code `newfc` wird dabei ausgelöst. Dabei kann es sich um einen internen Code aus dem Menü handeln, z.B. `SCMP`, oder einen Transaktionsaufruf wie z.B. `/NFB01` oder `/OMM02`. Die Enter-Taste kann durch "/0" angegeben werden.

Spezielle Controls	`On control="controlname" item="itemnid" event="eventno" Fcode="newfc" Process="script"` Behandlung eines einfachen Ereignisses (Buttonklick, Auswahl Menüeintrag) in speziellen Controls (z.B. "Objektdienste"). Bitte verwenden Sie die "Input Recording" Funktion zum Aufzeichnen der korrekten Angaben (control, item und event), die identisch sind mit den Angaben in "Enter".
Tips &Tricks	Kombinationen von Funktionstasten mit Ctrl und Shift können als Zahlen zwischen 13 und 48 eingegeben werden. Dabei entspricht **Shift** = +12 **Ctrl** = +24 Die Enter-Taste kann durch **/0** dargestellt werden. Shift+F1 /13 Ctrlt+F1 /25 Shift+Ctrl+F1 /37 Shift+F2 /14 Ctrlt+F2 /26 Shift+Ctrl+F2 /38 Shift+F3 /15 Ctrlt+F3 /27 Shift+Ctrl+F3 /39 Shift+F4 /16 Ctrlt+F4 /28 Shift+Ctrl+F4 /40 Shift+F5 /17 Ctrlt+F5 /29 Shift+Ctrl+F5 /41 Shift+F6 /18 Ctrlt+F6 /30 Shift+Ctrl+F6 /42 Shift+F7 /19 Ctrlt+F7 /31 Shift+Ctrl+F7 /43 Shift+F8 /20 Ctrlt+F8 /32 Shift+Ctrl+F8 /44 Shift+F9 /21 Ctrlt+F9 /33 Shift+Ctrl+F9 /45 Shift+F10 /22 Ctrlt+F10 /34 Shift+Ctrl+F10 /46 Shift+F11 /23 Ctrlt+F11 /35 Shift+Ctrl+F11 /47 Shift+F12 /24 Ctrlt+F12 /36 Shift+Ctrl+F12 /48 Beispiel: `On "/39" Fcode="/0" Process="warnung.txt"` Wirkung: Falls der Benutzer Ctrl+Shift+F3 drückt, wird die Enter-Taste ausgelöst und das Skript warnung.txt prozessiert.

7.26. Openfile

Funktion	Mit der `OpenFile`-Anweisung können Sie eine Datei öffnen und anschließend ihren Inhalt lesen (`ReadFile`) oder schreiben (`AppendFile`).

Beispiele	`OpenFile "X:GuiXTFiles\MatDes.txt"`
	
	Die Datei wird zum Lesen geöffnet.
	
	`OpenFile "X:GuiXTFiles\MatDes.txt" -Output Delimiter=";"`

Format	`OpenFile "Dateiname"` Weitere Optionen:

	`-Output` Die Datei wird gelöscht, falls vorhanden, und kann mit `AppendFile` beschrieben werden. `-Append` Die Datei bleibt bestehen, mit `AppendFile` können weitere Sätze hinzugefügt werden. `-UTF8` In Verbindung mit -Output: schreibt den UTF8 Header vor dem ersten Satz `-nocache` Der GuiXT Cache wird beim Lesen der Datei nicht verwendet. Andernfalls werden Dateien aus dem SAP Web Repository und Dateien in UNC notation "\\server\..." zunächst im GuiXT Cache gesucht, falls diese Option im GuiXT Profile angeschaltet ist. `Delimiter="x"` Das Zeichen x (z.B. ; oder ein Leerzeichen) wird bei `ReadFile` als Trennzeichen zwischen den einzelnen Feldern des eingelesenen Dateisatzes interpretiert. Default ist das Tabulatorzeichen. `-nodelimiter` Es wird kein Trennzeichen verwendet, d.h. bei `ReadFile` wird jeweils der gesamte Satz in die erste angegebene Variable eingelesen
Tipps & Tricks	• Das Öffnen oder Schließen einer Datei bezieht sich auf den jeweiligen SAP Modus • Das explizite Öffnen der Datei mit `OpenFile` ist nicht unbedingt nötig, da die Datei bei dem ersten `ReadFile` oder `AppendFile` automatisch geöffnet wird. Es ist aber empfehlenswert, die Datei explizit zu öffnen, da sonst ein fehlender `CloseFile` dazu führt, daß `ReadFile` bei dem nächsten Mal keinen Satz einliest. • Ein ausführliches Beispiel finden Sie in im Special Topic " Die Behandlung von Dateien in InputScripts ".

7.27. Parameter

Funktion	Mit der `Parameter`-Anweisung können Sie in einem InputScript Parameter definieren, die bei Aufruf des InputScript mit Werten versehen werden.

Beispiele	`Parameter Betrag`
	
	`Parameter Curr "DEM"`
	
	Es werden zwei Parameter definiert: `Betrag` und `Curr`. Der Parameter `Curr` erhält den Wert `"DEM"`, falls kein `using` dazu im Aufruf des InputScript angegeben ist.
	
	Im InputScript werden die Parameter mit dem Namen `&[Name]` angesprochen, also z.B. `&[Betrag]` oder `&[Curr]`.

Format	`Parameter Name`
	
	`Parameter Name "Defaultwert"`

Tips	• Sie können abfragen, ob ein Parameter bewertet wurde (beim Aufruf durch `using`

&Tricks	oder als Defaultwert):
	`if U[Name]`
	• Ebenfalls kann auf bestimmte Werte abgefragt werden
	`if U[Name=Wert]`
	Beispiel:
	`if not U[Curr=USD] and not U[Curr=DEM]`
	`  Return "Währung &[Curr] hier nicht unterstützt"`
	`endif`
	• Vergleichen Sie auch das InputAssistant Tutorial 3 Parameter und Eingabefelder.

7.28. Radiobutton

Funktion	Mit der `RadioButton` Anweisung können Sie Auswahlknöpfe (radio buttons) definieren, die dem Wert einer Variablen V[...] entsprechen. Alle Auswahlknöpfe mit dem gleichen Variablennamen werden als Gruppe betrachtet, d.h. wenn der Benutzer auf einen Knopf der Gruppe klickt, werden die anderen deaktiviert. Bitte beachten: Zusätzlich gibt es eine `RadioButton` Syntax, um ein Eingabefeld durch eine Reihe von Auswahlknöpfen zu ersetzen. Siehe GuiXT Dokumentation Radionbutton.

Beispiel	`// Vorschlagswert für Verkaufsbüro setzen`
	`if not V[SelOffice]`
	`Set V[SelOffice] "60"`
	`endif`
	`// Auswahl Büro`
	`Offset (8,0)`
	`Box(0,0) (4,30) "Zuständiges Verkaufsbüro"`
	`RadioButton(1,1) "Boston" Name="SelOffice" Value="60"`
	`RadioButton(2,1) "Paris" Name="SelOffice" Value="62"`
	`RadioButton(3,1) "Roma" Name="SelOffice" Value="80"`
	Hierdurch werden 3 Auswahlknöpfe definiert. Die Variable `V[SelOffice]` erhält entsprechend einen der Werte "60", "62", "80", je nach Auswahl eines Knopfes durch den Benutzer.

Format	`RadioButton (Zeile,Spalte) "text" Name="..." Value="..."`
Zusatzoptionen	`-readonly` Nur Ausgabe
Tipps & Tricks	Bitte beachten Sie auch die InputAssistant Tutorials 3 Parameter und Eingabefelder und 4 Felder auf Reitern.

7.29. Readfile

Funktion

Mit der **ReadFile**-Anweisung können Sie einen Satz aus einer Datei in GuiXT-Variablen einlesen und dann in einem InputScript weiterverarbeiten.

Beispiel

```
ReadFile"X:GuiXTFiles\MatDes.txt" s_matnr s_text s_unit s_price
```

Es wird ein Satz aus der Datei gelesen und in einzelne Felder unterteilt. Trennzeichen ist dabei das unter **OpenFile**angegebene Trennzeichen (**Delimiter=**, Default-Trennzeichen ist das Tabulatorzeichen). Die einzelnen Felder werden der Reihe nach in die angegebenen GuiXT-Variablen gefüllt.

Wurde aus der Datei etwa der Satz

1000001;Schraube;ST;0,02

eingelesen, so sind die Variablen anschließend wie folgt gefüllt:

&[s_matnr] = 10000120 **&[s_text]** = Schraube **&[s_unit]** = ST **&[s_price]** = 0,02

Format

```
ReadFile "filename" var1 var2 var3 ...
```

Die Angabe von maximal 60 Variablen ist möglich (vor GuiXT Version 2009 Q1 2: 20 Variablen).
Falls mehr Werte angegeben sind als in dem Dateisatz gefunden wurden, werden die restlichen Variablen auf Space gesetzt.

Falls die Datei keinen weiteren Satz enthält (Dateiende), werden alle angegebenen Variablen auf Space gesetzt.

Zusatzoptionen

-StripQuotationMarks	Falls Werte in Hochkommata eingeschlossen sind, werden diese automatisch entfernt. Beispiel: `ReadFile"-StripQuotationMarks""X:GuiXTFiles\MatDes.txt" s_matnr s_text s_unit s_price`

Tipps & Tricks

- Die Dateiendeabfrage kann in einem InputScript z.B. durch
  ```
  if not V[s_matnr]
  goto script_ende
  endif
  ```
  geschehen
- Ein ausführliches Beispiel finden Sie im "Special Topic" Die Behandlung von Dateien in InputScripts.

7.30. RemoveFile

Funktion	Mit der `RemoveFile`-Anweisung können Sie eine Datei löschen.
Beispiel	`RemoveFile "X:GuiXTFiles\temp.txt"` Die Datei wird gelöscht.
Format	`RemoveFile "filename"`
Tips &Tricks	• Durch OpenFile mit der Option `-output` wird der Dateiinhalt gelöscht, während die Datei im Dateiverzeichnis erhalten bleibt. RemoveFile löscht auch den EIntrag im Dateiverzeichnis

7.31. RenameFile

Funktion	Mit der `RenameFile`-Anweisung können Sie eine Datei umbenennen.
Beispiel	`RenameFile oldname="C:\images\p100.gif" newname="C:\images\logo.gif"` Die Datei `"C:\images\p100.gif"` wird unter Namen `"C:\images\logo.gif"` kopiert.

Format	`RenameFile oldname="sourcefile" newname="destinationfile"`
	Die Datei wird umbenannt. Falls eine Datei mit dem neuen Namen schon existiert, wird keine Aktion durchgeführt.
	Durch
	`if Q[ok]`
	`...`
	`endif`
	können Sie abfragen, ob die Operation erfolgreich war.

7.32. Retry on Error

Funktion	Mit der `Retry on error`-Anweisung können Sie erreichen, daß bei Ausgabe einer Fehlermeldung in dem aktuellen Bild wiederholt die "Enter"-Taste gedrückt wird, bis die Fehlermeldung verschwunden ist. Das ist speziell dann sinnvoll, wenn Sie in einem InputScript nach Abschluss einer Änderungs- oder Hinzufüge-Transaktion eine weitere Transaktion aufrufen, die auf das Verbuchen der ersten Transaktion warten muss. Das System gibt dann zunächst, solange die Verbuchung der Änderungen noch nicht abgeschlossen ist, eine Fehlermeldung zurück. Falls Sie bei der entsprechenden `Screen` Anweisung `Retry on error` verwenden, wird automatisch nach einer gewissen Zeit noch einmal die "Enter"-Taste gedrückt. Das wiederholte Drücken der "Enter"-Taste erfolgt bis zu 20 mal in jeweils erhöhten Zeitintervallen: 0.3, 0.6, 0.9, 1.2, ... Sekunden. Falls danach immer noch eine Fehlermeldung erscheint, wird die Verarbeitung so fortgesetzt, als wäre ohne "`Retry on error`" eine Fehlermeldung erfolgt. Sie benötigen bei Verwendung von "`Retry on error`" keine zusätzlichen `Screen`

	Anweisungen.
Beispiel	`Retry on error`
Format	`Retry on error`
Tips &Tricks	Falls Sie das InputScript "hell" prozessieren ("Visible auto processing"), wird die Option "`Retry on error`" ignoriert. Ein Anwendungsbeispiel finden Sie im Tip 15 Processing multiple transactions in one script.

7.33. ReplaceText

Funktion	Mit der `ReplaceText`-Anweisung können Sie in einer Langtextvariablen einen String durch einen anderen ersetzen.
Beispiele	`ReplaceText "t1" from="<konto>" to="1245678"` In der Langtextvariablen `t1` wird der String `"<konto>"` an allen Stellen durch `"1245678"` ersetzt.

Format	`ReplaceText "textvar" from="string1" to="string2"`
	
	In der Langtextvariablen `textvar` wird der unter `from=` genannte String an allen Stellen durch den unter `to=` gennanten String ersetzt. Sie können anschliessend durch `if Q[ok]` abfragen, ob der String mindestens einmal gefunden wurde.
	
	Statt `from=` können Sie auch `fromHexCode=` angeben; ebenso statt `to=` `toHexCode=`. Damit können Sie beliebige in hexadezimaler Schreibweise angegebene Codes ersetzen lassen. Beispiele:
	
	`// set newline code`
	`ReplaceText "t1" from="<newline>" toHexCode="0d0a"`
	
	`// delete utf8-header`
	`ReplaceText "t1" fromHexCode="efbbbf" to=""`

7.34. Return

Funktion	Mit der `Return`-Anweisung können Sie aus dem Prozessieren eines InputScript zu dem aufrufenden Bild zurückkehren.	
Beispiel	`Return "Bei Schnellbuchung ist nur Währung DEM vorgesehen"`	

Format	`Return`
	`Return "Nachrichtentext"`
	
	`Return "Nachrichtentext" "Titel"`
	
	Nach der Rückkehr wird die Nachricht `"Nachrichtentext"` mit dem Titel `"Titel"` angezeigt.	

Optionen	`-StatusLine`	Die Nachricht wird in der Statuszeile angezeigt. Dabei kann unter Umständen eine Nachricht der SAP-Anwendung überdeckt werden.

	Falls der Nachrichtentext mit "S: " oder "E: " oder "W: " beginnt, wird die Nachricht als Information oder Fehlermeldung oder Warnung angezeigt. Beispiel: `Return "E: Betrag &V[Betrag] zu hoch, Limit ist 50.000" -StatusLine` • Der Title= Parameter wird in diesem Fall ignoriert • Die Option -statusline wird in "Sapgui for HTML" Umgebung (ITS) automatisch angenommen
Tips &Tricks	• Variablen können wie üblich in den Nachrichtentext oder Titel eingestreut werden • Durch `\n` können Zeilenvorschübe im Nachrichtentext eingefügt werden, durch `\t` ein Tabulator

7.35. Returnvalues

Funktion	Mit der `Returnvalues`-Anweisung können Sie aus einem mit `"/O...,process=... "` geöffneten neuen Modus Werte zurück in den ursprünglichen Modus übertragen. Es kann sich bei den Zielfeldern um Eingabefelder oder Variablen handeln. Ebenfalls möglich ist das Auslösen eines Funktionscodes und Start eines InputScript mit Parameterübergabe.

Beispiele	`Returnvalues "Material:&V[mat];Menge:&V[mng]"`
	In das Eingabefeld `Material` der ursprünglichen Transaktion wird der Wert der Variablen `V[mat]` aus dem /O-Modus gesetzt, entsprechend in das Feld `Menge` der Wert von `V[mng]`.
	`Returnvalues "V[mat]:&V[matnr]"`
	In die Variable `V[mat]` des ursprünglichen Modus wird der Wert der Variablen `V[matnr]` aus dem /O-Modus gesetzt.
	`Returnvalues "U[MATNR]:&V[matnr];OK:/8,process=next.txt"`
	In der ursprünglichen Transaktion wird die F8-Taste gedrückt und das InputScript "`next.txt`" angestossen. Als Parameter `U[MATNR]` wird der Wert der Variablen `V[matnr]` aus dem /O-Modus mitgeggeben.

Format	`Returnvalues string` Der Aufbau des Strings ist identisch mit dem bei `Image` beschriebenen String für den Parameter `Input="..."`, und identisch mit dem dort beschrieben Aufruf von GuiXT aus externen Programmen.
Tips & Tricks	• Es ist auch möglich, eine Reihe von `Returnvalues` Anweisungen zu verwenden für einzelne Variablen, oder auch für den Inhalt einer Textbox. Beispiel: Sie verwenden das folgende GuiXT Script, wobei xxxx ein beliebiger Transaktionscode

ist, zum Testen z.B. SE37:

```
Pushbutton (toolbar) "Test Returnvalues" "/Oxxxx" process="tv.txt"

Offset (10,0)
Text (0,0) "&V[t1]"
Text (1,0) "&V[t2]"

Textbox (2,1) (8,100) name="txt1"
```

Das InputScript "tv.txt" kann leer sein. In der gerufenen Transaktion dann folgendes GuiXT Script:

```
Offset (15,1)
Box (0,0) (10,42) "Test Returnvalues"
InputField (1,0) "Line1" (1,8) size=30 name=t1
InputField (2,0) "Line2" (2,8) size=30 name=t2
Textbox (3,0) (9,40) name="txt1"

Pushbutton (toolbar) "return values" process="rv.txt"
```

Und das InputScript "rv.txt":

```
Returnvalues "V[t1]: &V[t1];"
Returnvalues "V[t2]: &V[t2];"

Returnvalues "Text[txt1]:&Text[txt1];"

Returnvalues "OK:/0"

Enter "/i"
```

7.36. SearchHelp

Funktion	Mit der `SearchHelp`-Anweisung können Sie eine Suchhilfe (F4 Matchcode) für existierende SAP-Felder und Tabellenspalten definieren oder den existierenden Matchcode überschreiben.
Beispiel	`SearchHelp F[Externer Buchungskreis] techname="BKPF-BUKRS"`

Format	`SearchHelp F[Eingabefeld] [optionen] // Eingabefeld`
	`SearchHelp [Spalte] [optionen] // Tabellenspalte`
	`SearchHelp [Tabelle,Spalte] [optionen] // Tabellenspalte`

Option	`-none` — Es wird keine Suchhilfe angeboten.

Weitere Optionen	`TechName=`
	`shname=`
	`shselname1=`
	`shselvalue1=` — Die Bedeutung der einzelnen Optionen ist identisch mit den

	shselname2= shselvalue2= shselname3= shselvalue3= shname1= shdest1= shname2= shdest2= shname3= shdest3= SearchHelpProcess=	gleichlautenden Optionen in InputField

7.37. SelectDirectory

Funktion	Mit der SelectDirectory-Anweisung können Sie einen Standarddialog zur Auswahl eines Verzeichnisses aufrufen.
Beispiel	`SelectDirectory name="MM_dir"`
Format	`SelectDirectory name="dirname"` Der Dialog zur Auswahl des Verzeichnisses wird aufgerufen. Die Variable `V[dirname]` enthält danach den ausgewählten Verzeichnisnamen. Durch `if Q[ok]` können Sie nach `SelectDirectory` abfragen, ob der Benutzer tatsächlich ein Verzeichnis ausgewählt hat.

Optionen		
	title=	Angabe eines Titels
	directory=	Das angegebene Verzeichnis wird als erstes zur Auswahl angeboten

7.38. SelectFile

Funktion	Mit der SelectFile-Anweisung können Sie einen Standarddialog zur Dateiauswahl aufrufen.
Beispiel	`SelectFile name="MM_fnam"`
Format	`SelectFile name="filename"` Der Dialog zur Dateiauswahl wird aufgerufen. Die Variable V[filename] enthält danach den ausgewählten Dateinamen. Durch if Q[ok] können Sie nach SelectFile abfragen, ob der Benutzer tatsächlich eine Datei ausgewählt hat.

Optionen		
	title=	Angabe eines Titels für die Dateiauswahl
	directory=	Das angegebene Verzeichnis wird als erstes zur Auswahl angeboten
	filter=	Angabe eines oder meherer Bedingungen für den Dateinamen, durch ";" voneinander getrennt. Beispiel: `filter="*.gif;*.jpg;*.bmp"`
	-save	Statt des "Öffnen" Dialogs wird der "Sichern" Dialog aufgerufen.

7.39. Screen

Funktion

Mit der `Screen`-Anweisung leiten Sie in einem InputScript die automatische Verarbeitung dieses Bildes ein.

Während der InputAssistant die Transaktion intern durchführt, sucht er im InputScript jeweils die nächste passende Screen-Anweisung und führt die entsprechenden Anweisungen aus. Falls er keine passende Screen-Anweisung findet, wird dem Benutzer das aktuelle Bild angezeigt, und er kann manuell weiterarbeiten.

Beispiel

```
Screen SAPMF05A.0300
```

Hier beginnt die Verarbeitung von Bild **0300** zu Programm **SAPMF05A**.

Format

```
Screen programm.bildnummer
```

Tips &Tricks

- Bei geschachtelten Masken muß nur das oberste Bild (also das, in dem der Benutzer Werte eingeben kann) angegeben werden
- Die Bildnummer immer 4-stellig angeben
- In einem InputScript ist es sehr verwirrend und keinesfalls zu empfehlen, `Screen`-Anweisungen innerhalb von `if ... endif` zu benutzen. Schlechtes Beispiel:

```
Screen S1
   Set V[x] "a"
   Enter

   if V[x=b]

      Screen S2
        Enter "xx"
   else

      Screen S2
        Enter "yy"

   endif
```

Nehmen wir an, es erscheint zunächst `Screen S1`, dann `Screen S2`. Wird nun Enter "xx" oder Enter "yy" ausgeführt? Tatsächlich wird im Beispiel oben Enter "xx" ausgeführt, da GuiXT nach Verarbeiten von Screen S1 eine passende Screen-Anweisung zu dem Screen S2 sucht, ohne eventuell noch offene if-Schachtelungen früherer Screen-Blöcke zu berücksichtigen.
Stattdessen bitte immer die if-endif-Schachtelung pro Screen-Anweisung abschliessen:

```
Screen S1
   Set V[x] "a"
   Enter

Screen S2
   if V[x=b]
      Enter "xx"
   else
```

```
      Enter "yy"
   endif
```

Falls nötig, können Sie auch mit goto/label arbeiten:

```
Screen S1
   Set V[x] "a"
   Enter

   if V[x=b]
     goto l_b
   endif

Screen S2
   Enter "xx"
   goto l_weiter

label l_b
Screen S2
   Enter "yy"

label l_weiter
```

7.40. Set

Funktion

Mit der `Set`-Anweisung können Sie Werte in Eingabefelder setzen. Die `Set`-Anweisung ist insbesondere zum Gebrauch in einem InputScript (Einbildtransaktionen) gedacht.

Im Unterschied zu `Default` prüft `Set` nicht, ob das Eingabefeld leer ist, sondern überschreibt einen eventuell existierenden Wert. In normalen GuiXT-Skripten ist `Set` kaum verwendbar, da der ursprünglich anzuzeigende Wert schon bei Anzeige des Bildes überschrieben wird.

Ausser Eingabefeldern können Sie durch `Set` auch intern benutzten Variablen einen Wert zuweisen: `Set V[Varname] "Wert"`.

Beispiel

```
Set F[Auftragsart] "BV"
```

Das Eingabefeld Auftragsart erhält den Wert `"BV"`.

Häufig wird als Wert ein Parameter des InputScript oder ein vorher definiertes Eingabefeld genannt, z.B.

```
Parameter AufArt

...

Set F[Auftragsart] "&U[AufArt]"
```

Format

`Set F[Eingabefeld] "Wert"`	setzt einen Wert für ein Eingabefeld.
`Set V[Varname] "Wert"`	setzt eine globale Variable.
`Set V[gnm*] "Wert"`	setzt einen Wert für alle globalen Variablen bei

	denen der Name mit "gnm*" beginnt.
`Set V[varname](x-y) "Wert"`	setzt einen Wert in die Stellen x bis y. Falls die bisherige Länge der Variablen kürzer ist als x, wird mit Leerzeichen aufgefüllt.

`Set Text[xxx] "Wert"`	als Kurzschreibweise für:
	`Set V[y] "Wert"`
	`CopyText fromString="y" toText="xxx"`
	Analog wird `&Text[xxx]` in Strings durch den aktuellen Inhalt des Langtextes xxx ersetzt.

`Set list(row,col] "Wert"` `Set list(row,col,len] "Wert"`	Nur in ABAP-Listanzeige möglich. Der angegebene Wert wird in die Liste geschrieben, und zwar in Zeile `row` und Spalte `col`, wobei die Zählung bei 0 beginnt und sich auf die aktuell angezeigte Seite bezieht. Optional kann durch `len` die gewünschte Länge des Strings mitgegeben werden. Hinweis: Die Systemvariable V[_listfirstvisiblerow] enthält die absolute Nummer der ersten auf der Seite angezeigten Zeile, bei 1 beginnend.

Zusatzoptionen	
`-stringlength`	Die Länge des Strings (in Bytes) wird in der Variablen zur Verfügung gestellt
`-charcount`	Die Anzahl Zeichen des Strings wird in der Variablen zur Verfügung gestellt
`search=`	Angabe eines Suchstrings. Wird der String gefunden, so wird das danach angegebene Wort übertragen. Beispiel: `Set V[Belegnr] "&V[_message]" Search="Beleg"` Falls die Systemvariable `V[_message]` den Inhalt "Es wurde der Beleg 10004003 erzeugt" hat, so wird der Variablen `V[Belegnr]` der Wert "10004003" zugewiesen. Bei der Suche wird Gross/Kleinschreibung ignoriert. Durch `if Q[ok]` kann man abfragen, ob der String gefunden wurde.
`pattern=`	Angabe einer Suchmaske, welche die Zeichen • '%' beliebiger String bis zum nächsten passenden Zeichen

		der Suchmaske • '_' einzelnes Zeichen enthalten kann. Es werden von links nach rechts alle Zeichen in die Variable übertragen, die einem "%" oder "_" entsprechen. Wenn die angegebene Suchmaske nicht mehr mit dem String übereinstimmt, werden keine weiteren Zeichen übertragen. Durch `if O[ok]` kann man abfragen, ob die Suchmaske vollständig den angegebenen String beschrieben hat. Beispiele: `SetV[x] "02.05.2007" pattern="__.__.____"` // ok V[x] = "02052007" `Set V[x] "[abcd]" pattern="[%]"` // ok V[x] ="abcd" `Set V[x] "12" pattern="%kg"` // nicht ok V[x]= "12" `Set V[x] "Beleg 400000087 hinzugefügt" pattern="Beleg % hinzugefügt"` // ok V[x]= "400000087"
	`-uppercase`	Der Wert wird in Grossbuchstaben umgesetzt
	`-lowercase`	Der Wert wird in Kleinbuchstaben umgesetzt
	`-unpack`	Ein Wert, der aus einem Funktionsbaustein/BAPI als gepackter Dezimalwert geliefert wurde, wird als normale Ziffernreihe übertragen.

Rechenoperationen und Zusatzoptionen		`Set` erlaubt die Angabe von Rechenoperationen +, -, *, / mit jeweils zwei Operanden. Das Ergebnis der Rechenoperation wird in aufbereiteter Form in die Variable gestellt. Die beiden Operanden können als Direktwerte (z.B. 1 oder 5830) oder als Inhalte von Variablen ("&V[...]") angegeben werden. Beispiele:
		`Set V[Wert] 5830 / 365`
		`Set V[Wert] "&V[Gesamt]" / 365`
		`Set V[Wert] "&V[Gesamt]" / "&V[Tage]"`
		`Set V[Summe] "&V[Betrag1]" +"&V[Betrag2]"`
		`Set V[Index] "&V[Index]" +1`
		Das Ergebnis wird kaufmännisch gerundet und mit 2 Dezimalstellen in die Ergebnisvariable gestellt. Dabei wird bei Anmeldung mit Sprache Englisch der Punkt, sonst das Komma zur Abtrennung der Dezimalstellen verwendet (siehe auch die Zusatzoptionen und die Anweisung `DecimalSeparator`). Ganzzahlige Werte werden ohne Dezimalstellen dargestellt.

	`decimals=`	Angabe der Anzahl gewünschter Dezimalstellen zur Aufbereitung des Ergebnisses (0,1,2,3,...). Auch ganzzahlige Werte werden mit der angegebenen Anzahl

	von Dezimalstellen dargestellt.
`decimalseparator=`	Angabe des Trennzeichens für die Dezimalstellen: "," oder "." . Dieses wird sowohl bei der Interpretation der Operanden als auch bei der Darstellung des Ergebnisses berücksichtigt.
`decimalseparator=UserDefault`	Der im Benutzerstammsatz abgelegte Default (Punkt oder Komma) wird benutzt. Beim ersten Verwenden dieser Anweisung wird der Defaultwert über einen Remote Function Call (RFC) eingelesen. Es ist daher ein gültiger RFC Benutzer im GuiXT-Profile erforderlich.
`groupseparator=`	Angabe eines Trennzeichens für die Tausendergruppen, meist "," oder ".". Es wird zur Darstellung des Ergebnisses verwendet.
`groupseparator=UserDefault`	Der im Benutzerstammsatz abgelegte Default (Punkt oder Komma) wird benutzt. Beim ersten Verwenden dieser Anweisung wird der Defaultwert über einen Remote Function Call (RFC) eingelesen. Es ist daher ein gültiger RFC Benutzer im GuiXT-Profile erforderlich.

Beispiel

```
Set V[x] 246824717 Decimals=3 DecimalSeparator="." GroupSeparator=","
```

Die Variable V[x] erhält den Wert "3,526,067.286". Das exakte Ergebnis ist "3526067,285714..."; es wird auf 3 Dezimalstellen gerundet und gemäss den Optionen aufbereitet.

Tipps & Tricks

- Das Ankreuzen einer Checkbox ist ebenfalls mit `Set` möglich. Als Werte sind `"X"` oder `" "` möglich, oder entsprechend die Angabe eines Parameters, der einen dieser Werte besitzt.
- Analog aktivieren Sie durch `Set` einen Radiobutton. Dabei ist es nicht nötig, die übrigen Radiobuttons der gleichen Gruppe zu deaktivieren, das geschieht automatisch bei dem Aktivieren eines einzelnen Radiobuttons. Zum Beispiel werden durch

  ```
  Set R[Dokumentation]"X"
  ```

  in Transaktion SE38 die weiteren Radiobuttons der gleichen Gruppe, `R[Quelltext]`, `R[Varianten]`, `R[Attribute]`, `R[Textelemente]`, auf nicht aktiv gesetzt.

	• Bitte lesen Sie das InputAssistant Tutorial zur Anwendung von `Set`.

7.41. SetActiveTab

Funktion	Mit der `SetActiveTab`-Anweisung können Sie den aktiven Reiter festlegen. **Bitte beachten:** Das ist nur möglich bei "lokal" prozessierten Reitern, etwas in der SAP-Transaktion MIGO. Bei den sehr viel häufigeren Reiten, bei denen ein Klick nicht lokal, sondern auf dem Applikationsserver behandelt wird, ist stattdessen ein "Enter P[tabname]" nötig, um zu dem gewünschten Reiter zu wechseln.
Beispiel	`SetActiveTab P[Lieferant]`
Format	`SetActiveTab P[tabname]` Setzt den aktiven Reiter.

7.42. SetCursor

Funktion	Mit der `SetCursor`-Anweisung können Sie den Cursor in ein bestimmtes Feld, in eine Tabellenzelle oder auf eine bestimmte Position in einer Liste setzen.

Beispiele	`SetCursor F[Name]`
	`SetCursor cell[Table,Material,5]`
	`SetCursor (4,1)`

Format	`SetCursor F[Feldname]`
	Setzt den Cursor in das angegebene Feld
	`SetCursor cell[Tabelle,Spaltenname,Zeile]`
	Setzt den Cursor in die Tabellenzelle.
	`SetCursor (Zeile,Spalte)`
	Listanzeige: Setzt den Cursor in eine Zeile/Spalte.

Optionen	`Offset=x` Der Cursor wird innerhalb des Feldes (oder der Tabellenzelle) mit x Zeichen vom linken Rand positioniert
Tips & Tricks	• Folgendermaßen können Sie den Cursor mit einer zugewiesenen Tastenkombination (z.B. Strg+F12) in eine absolute Position setzen, z.B. in eine bestimmte Spalte und Zeile einer Tabelle:

```
//GuiXT script:

Pushbutton (toolbar) "Material column" process="va01_setcursor2.txt"
"Ctrl+F12"

if V[NewCursorPosition]

   SetCursor "&V[NewCursorPosition]"

   Set V[NewCursorPosition] ""

endif

// InputScript "va01_setcursor2.txt":
```

```
Set V[NewCursorPosition] "cell[All items,2,&V[_tabrow]]"
```

- Folgendermaßen setzen Sie den Cursor bei einem Enter-Event zwei Spalten nach links:

```
//GuiXT script:

if V[NewCursorPosition]

   SetCursor "&V[NewCursorPosition]"

   Set V[NewCursorPosition] ""

endif

On Enter Process="va01_skip2cols.txt"

//InputScript "va01_skip2cols.txt":

Set V[NextColumn] "&V[_tabcol]" + 2

Set V[NewCursorPosition] "cell[All Items,&V[NextColumn],&V[_tabrow]]"
```

- Wenn Sie den Cursor auf einem bestimmten Pushbutton haben möchten, nachdem ein InputScript ausgeführt wurde, z.B.: Button1 Button2 Button3 sind im Bild. Der Anwender klickt einen Button1, führt dadurch ein InputScript aus und kommt dann zum Hauptbild zurück mit den drei Pushbuttons. Der Cursor soll nun auf dem zuletzt angeklickten Button1 sein.
  SetCursor erlaubt momentan nicht, den Cursor direkt auf einen Button zu setzen. Sie können aber die Koordinaten des Pushbuttons in runden(!) Klammern verwenden. In diesem Fall überprüft GuiXT gar nicht, ob an dieser Stelle ein Eingabefeld liegt. Beispiel:

```
Pushbutton (10,1) "Button1" process="t1.txt"
Pushbutton (12,1) "Button2" process="t2.txt"
Pushbutton (14,1) "Button3" process="t3.txt"

if V[xx_status=1]

  SetCursor (10,1)

endif

if V[xx_status=2]

  SetCursor (12,1)

endif

if V[xx_status=3]

  SetCursor (14,1)

endif
```

(Sie setzen xx_status in Ihrem InputScript.)

7.43. SetLeftTab

Funktion	Mit der `SetLeftTab`-Anweisung können Sie bei "lokal" prozessierten Reitern den links als erstes angezeigten Reiter festlegen.
Beispiel	`SetLeftTab P[Lieferant]`
Format	`SetLeftTab P[tabname]` Definiert den ersten angezeigten Reiter.

7.44. Sort

Funktion	Mit der `Sort`-Anweisung können Sie die Werte einer fortlaufend nummerierten Anzahl von Variablen umsortieren. Auch mehrere Variablenleisten können simultan sortiert werden, beispielsweise wenn Sie eine tabellenartige Darstellung mit GuiXT Variablen implementieren.

Beispiele	`Sort "mx_matnr_"`
	Aufsteigende Sortierung der Werte in den Variablen V[mx_matnr_1], V[mx_mantr_2], V[mx_matnr_3] ,...
	`Sort "mx_tab_%row_%col" -descending rowcount=84 colcount=12 orderby=3`
	Absteigende Sortierung einer Tabelle aus Variablen V[mx_tab_1_1], V[mx_tab_1_2], V[mx_tab_1_3], ... bis V[mx_tab_84_12].

Format	`Sort "varname"`
	Sortieren der Variablen V[varname1], V[varname2],..
	`Sort "varname_%row_%col"`
	Sortieren der Zeilen einer Tabelle aus Variablen, wobei für "%row" die Zeilennummer und für "%col" die Spaltennummer eingesetzt wird.

Zusatzoptionen	`-descending` Absteigende Sortierung `rowcount=` Anzahl Zeilen. Falls nicht angegeben, ermittelt GuiXT die Anzahl der Zeilen aus der Anzahl existierender Variablennamen. `colcount=` Anzahl Spalten. Falls nicht angegeben, wird eine eindimensionale Leiste von Variablen angenommen. `orderby=` Nummer der Spalte, nach der sortiert werden soll. Falls nicht angegeben: erste Spalte
Zu beachten	• Die Spalten können Zeichenreihen (Strings), Zahlen oder Datumsangaben enthalten; die Sortierung erfolgt jeweils typgerecht. • Der Sort ist stabil, d.h. er ändert die Reihenfolge bei gleichen Werten nicht. Sie können daher, wenn die Sortierung nach mehr als einer Spalte gewünscht wird, die Sortierungen nacheinander durchführen, bei der letzten Sortierspalte beginnend.

7.45. Start

Funktion	Mit der `Start` Anweisung können Sie eine .exe oder .bat Datei ausführen.
Format	`Start "Dateiname"`
Weitere Optionen	`Parameters=` String mit Parametern `Directory=` Verzeichnisname

7.46. StatusMessage

Funktion	Mit der `StatusMessage`-Anweisung können Sie während der Verarbeitung eines InputScripts fortlaufend Meldungen über den Stand der Verarbeitung ausgeben. Sobald Sie das erste Mal `StatusMessage` aufgerufen haben, werden automatisch alle Systemmeldungen (auch Warnungen, Fehlermeldungen) dort ebenfalls protokolliert.

Beispiele	`StatusMessage Title="Materialstamm anlegen"`
	Hierdurch wird eine Listbox angezeigt mit dem Titel "Materialstamm anlegen". Alle Systemmeldungen sowie die selbst ausgegebenen Meldungen erscheinen dort parallel zur Verarbeitung des InputScripts.
	`StatusMessage AddString="Anlegen der Vertriebsdaten"`
	Fügt eine neue Zeile "Anlegen der Vertriebsdaten" in die Listbox ein.

Format	`StatusMessage`
Zusatzoptionen	`Title=` Angabe eines Titels. Der Titel sollte in der ersten `StatusMessage` angegeben werden. Er kann bei Bedarf später überschrieben werden. `AddString=` Text, der neu angezeigt werden soll `StopRequest=` Angabe eines Textes, z.B. "Abbruch der Verarbeitung". Dieser wird im Systemmenü der Listbox eingeblendet. Sobald der Benutzers auf den Menü-Eintrag klickt, wird ein interner Status "StopRequest" gesetzt, der im InputScript durch `if Q[StopRequest]` abgefragt werden kann, z.B. nach einem Enter oder innerhalb einer Dateiverarbeitung vor Einlesen eines neuen Satzes. Das InputScript kann dann bestimmte Endeverarbeitungen durchführen (z.B. Dateien schliessen) und die Verarbeitung insgesamt beenden. `Size=(Zeilen,Spalten)` Grösse der Listbox in Zeilen und Spalten. Falls einer der beiden Werte 0 ist, wird die Standardgrösse als Anzahl Zeilen bzw. Anzahl Spalten beibehalten. `Position=` Position der Listbox, in Zeilen und Spalten innerhalb des Bildschirms, oder symbolische Angabe. Beispiele:

	... Position=(10,40) ... Position=[Buchungskreis]+(2,10)
-ResetContent	Die Liste wird zurückgesetzt (0 Einträge)
-Remove	Die Listbox wird entfernt

toText=	Die Nachrichten werden in einen Langtext ausgegeben. Beispiel: Am Anfang eines InputScripts
	`StatusMessage toText="msg"`
	
	und am Endes eines InputScripts
	`if text[msg]`
	`  Message "&text[msg]"`
	`endif`

-setmsgid	Gibt bei SAP Meldungen auch die Nachrichten-Id (Arbeitsgebiet und Nummer) aus

SystemMessages=	Steuert die Aufnahme von Systemmeldungen in das StatusMessage Fenster. Sie können die Meldungstypen E (Error), W (Warning) und I (Information) angeben, oder On / Off. Beispiele:
	`SystemMessages=On  // Alle Systemmeldungen (default)`
	`SystemMessages="E,W,I"  // gleiche Bedeutung`
	
	`SystemMessages=Off  // Keine Systemmeldungen`
	`SystemMessages=""  // gleiche Bedeutung`
	
	`SystemMessages="W"  // Nur Warnungen`

Tips &Tricks	• Sie können bei `AddString=` Variablen in den Text einfügen, z.B. durch `StatusMessage Title="&V[_title]"` den Titel des dunkel prozessierten Bildes ausgeben.

7.47. Tab

Funktion	Mit der `Tab`-Anweisung können Sie die Tab-Reihenfolge der Felder und Tabellenspalten bestimmen. Sie können pro Feld oder Spalte angeben, in welches Feld bzw. in welche Spalte der Cursor bei Betätigen der Tab-Taste springen soll.
Beispiel	`Tab [Buchungskreis] [Buchungsdatum]`

Format	`Tab [Bildelement1] [Bildelement2]`
	`Tab [Tabellename,Spaltenname1] [Tabellenname,Spaltenname2]`
	
	Bei der Angabe von Spalten gilt darüber hinaus: Falls die neue Spalte nicht rechts von

	der bisherigen liegt, springt der Cursor automatisch in die nächste Zeile. Wenn am Ende der Tabelle eine neue Zeile nötig ist, löst die `Tab`-Anweisung automatisch "Enter" aus.
Tips &Tricks	• Falls Sie für eine Spalte die gleiche Spalte als tab-Ziel angeben, kann der Benutzer die betreffende Spalte leicht vertikal füllen. Entsprechend können Sie, falls z.B. die ersten drei Tabellenspalten für mehrere Zeilen ohne Mausbedienung eingegeben werden sollen, von der dritten Spalte in die erste springen (neue Zeile).

7.48. Table

Funktion	Mit der `Table`-Anweisung können Sie eine eigene Tabelle (SAP Table Control) hinzufügen.
Beispiel	`Table (10,20) (16,84) name="proj" title="Projektübersicht rows=20` Hierdurch wird eine neue Tabelle mit der Überschrift "Projektübersicht" hinzugefügt. Die Spalten der Tabellen werden anschliessend durch `Column`-Anweisungen definiert.
Format	`Table (Zeile1,Spalte1) (Zeile2,Spalte2) name="mytable" title="mytitle" rows="rowcount"` Eine neue Tabelle mit Überschrift "mytitle" wird hinzugefügt. Der interne Name ist "mytable". Die Gesamtzahl der Zeilen ist "rowcount". Die Anzahl der angezeigten Zeilen richtet sich nach der über die Koordinaten definierten Grösse. `Column`-Anweisungen werden automatisch der vorangehenden `Table`-Anweisung zugeordnet, falls nicht explizit dort mit `Table=` eine andere Tabelle angegeben ist.
Optionen	`fixedcolumns=` Anzahl der Spalten, die bei horizontalem Scrollen festgehalten werden `-rowselection` Eine oder mehrere Zeilen der Tabelle können selektiert werden `-singlerowselection` Eine Zeile der Tabelle kann selektiert werden `-columnselection` Eine oder mehrere Spalten der Tabelle können selektiert werden `-singlecolumnselection` Eine Spalte der Tabelle kann selektiert werden
Zuordnung der Variablen	Sowohl der Inhalt der Tabelle (Ein- und Ausgabe) als auch der Zustand (Blättern, Selektion von Zeilen und Spalten, Ändern von Spaltenbreiten oder Spaltenreihenfolge) wird in Variablen abgebildet, deren Namen jeweils mit dem gewählten Tabellennamen, z.B. "mytable", beginnt. Die Namen der Variablen für die Zelleninhalte beginnen mit "mytable.cell.", alle Statusvariablen beginnen mit "mytable.stat." **Inhalt der Tabellenzellen** Jede Tabellenzelle wird durch eine Variable V[tabname.cell.columnname.row] repräsentiert. Beispiel: `V[projects.cell.description.7]` ist die 7. Zeile, Spalte "description" in der Tabelle

"projects". Durch

```
Set V[projects.cell.*] ""
```

können Sie den gesamten Inhalte der Tabelle zurücksetzen. Falls der Zustand (Blättern, Zeilenselektion,...) zurückgesetzt werden soll, verwenden Sie

```
Set V[projects.stat.*9] ""
```

oder für Inhalt und Zustand:

```
Set V[projects.*] ""
```

Blättern Die Blätterfunktionen sind automatisch durch das SAP Table Control implementiert (Blättertasten, SAP-Ikonen, Scrollbar). Bei mehreren gleichzeitig angezeigten Tabellen beziehen sich die Blättertasten (Erste Seite, eine Seite vor/zurück, letzte Seite) auf die Tabelle, in der der Cursor steht, sonst auf die erste angezeigte Tabelle.

Die Variable V[tabname.stat.firstvisiblerow] enthält die Nummer der ersten angezeigten Zeile. Sie kann auch in einem Script verändert werden, um die Anzeige auf eine andere Zeile zu positionieren, zum Beispiel zur Implementierung einer Suchfunktion. Beispiel:

```
Set V[projects.stat.firstvisiblerow] 12
```

Dadurch wird die Tabelle ab Zeile 12 angezeigt. Wenn die Variable nicht gesetzt ist, wird der Wert 1 angenommen.

Die Variable V[tabname.stat.lastvisiblerow] enthält die Nummer der letzen angezeigten Zeile. Ihre Änderung in einem Script hat keine Auswirkungen.

Horizontales Blättern Horizontales Blättern durch die Spalten ist im SAP Table Control ebenfalls möglich. Die Variable V[tabname.stat.firstvisiblecolumn] enthält die Nummer der ersten angezeigten Spalte. Wenn die Variable nicht gesetzt ist, wird der Wert 1 angenommen.

Falls durch `fixedcolumns=` eine oder mehrere Spalten fixiert wurden, enthält V[tabname.stat.firstvisiblecolumn] die Nummer der ersten angezeigten Spalte innerhalb der scrollbaren Spalten, also z.B. bei 2 feststehenden Spalten mindestens die Nummer 3.

Reihenfolge der Spalten Die anfängliche Reihenfolge der Spalten ist durch die Reihenfolge der `column-`Anweisungen vorgegeben. Der Benutzer kann dann die Reihenfolge der Spalten verändern. Sie kann auch im Script dynamisch gesetzt werden. Die Variablen dazu heissen V[tabname.stat.columnnumber.columnname], z.B.
`V[projects.stat.columnnumber.description].`

Spaltenbreite Die anfängliche Breite der Spalten ist durch die `width=` Angabe der `column-` Anweisungen vorgegeben. Der Benutzer kann dann die Breite der Spalten verändern. Sie kann auch im Script dynamisch gesetzt werden. Die Variablen dazu heissen V[tabname.stat.columnwidth.columname], z.B.
`V[projects.stat.columnwidth.description].`

Zeilenselektion Bei einer selektierten Zeile ist die Variable

V[tabname.stat.rowselection.row] auf "X" gesetzt. Beispiel:

```
if V[projects.stat.rowselection.1=X]
```

fragt ab, ob die Zeile 1 selektiert wurde. Die Selektion einer Zeile kann auch durch ein Script gesetzt werden.

Spaltenselektion Bei einer selektierten Spalte ist die Variable V[tabname.stat.columnselection.columnname] auf "X" gesetzt. Beispiel:

```
if V[projects.stat.columnselection.description=X]
```

fragt ab, ob die Spalte "description" selektiert wurde. Die Selektion einer Spalte kann auch durch ein Script gesetzt werden.

Drucktasten in Tabellenspalten Durch die `column-`Anweisung kann können in einer Tabellenspalte Drucktasten dargestellt werden, die ein InputScript starten. Folgende Variablen sind dann gesetzt:

`V[_tabrow]` Nummer der ausgewählten Zeile in der Tabellenanzeige, beginnend mit 1 für die erste gerade angezeigte Zeile

`V[_tabrowabs]` Absolute Zeilennummer innerhalb der Tabelle

Suchhilfe (F4) in Tabellenspalten In der `column-`Anweisung können Sie den Tabellenspalten eine SAP-Suchhilfe zuordnen.

Cursorposition abfragen Mit folgenden Variablen kann die Cursorposition innerhalb der Tabelle abgefragt werden:

`V[_cursortabname]` Titel der Tabelle (title=)

`V[_cursortabtechname]` Technischer Name der Tabelle (name=)

`V[_cursorcolname]` Spaltenüberschrift

`V[_cursorcoltechname]` Technischer Name der Spalte

`V[_tabcol]` Spaltennummer

`V[_tabrow]` Zeilennummer in der Tabellenanzeige, beginnend mit 1 für die erste gerade angezeigte Zeile

`V[_tabrowabs]` Absolute Zeilennummer innerhalb der Tabelle

`V[_cursorrow]` Zeile, in der die Tabellenanzeige beginnt ("Zeile1" in `Table`-Anweisung)

`V[_cursorcol]` Spalte, in der die Tabellenanzeige beginnt ("Spalte1" in `Table`-Anweisung)

Setzen des Cursors in eine Tabellenzelle Durch SetCursor cell[tabname,columnname,row], zum Beispiel

```
SetCursor cell[project,Description,3]
```

Dabei ist die Zeilennummer die visuelle Zeilennummer in der Anzeige. Für columnname können Sie die Spaltenüberschrift oder den technischen Namen der Spalte verwenden.

Wenn Sie auf eine absolute Zeilennummer positionieren möchten, z.B. V[posrow], in Spalte "descr", geht das z.B. wie folgt:

```
if

V[posrow<&V[xtab.stat.firstvisiblerow]]orV[posrow>&V[xtab.stat.lastvisibl
erow]]
SetV[xtab.stat.firstvisiblerow]&V[posrow]
SetCursorcell[xtab,descr,1]
else
SetV[relposrow]&V[posrow] - &V[xtab.stat.firstvisiblerow]
SetV[relposrow]&V[relposrow] + 1
SetCursorcell[xtab,descr,&V[relposrow]]
endif
```

7.49. Textbox

Funktion	Mit der TextBox-Anweisung können Sie einen Text anzeigen und vom Benutzer ändern lassen.
Beispiele	**TextBox (10,20) (16,44) name="t1"** Es wird eine Textbereich mit dem linken oberen Eckpunkt **(10,50)** und dem rechten unteren Eckpunkt **(16,44)** eingeblendet. Der Text wird der Textvariable "t1" zugeordnet.
Format	**TextBox (zeile1,spalte1) (zeile2,spalte2) name="textname"**
Optionen	**-ReadOnly** Der angezeigte Text ist nicht änderbar **-noWordWrap** Kein automatischer Zeilenumbruch **Textfont=** Gewünschter Font. Default ist "Arial" . **Textheight=** Texthöhe in Pixel. Default ist 16. **Textweight=** Schriftdicke zwischen 1 (dünn) bis 9 (sehr dick). Default ist 5.
Tips&Tricks	• Mit der **CopyText** Anweisung können Sie Texte zwischen Dateien, Bildschirmbereichen und Textvariablen kopieren. • Anwendungsbeispiele für **TextBox** finden Sie in den Special Topics Texterfassung in InputScripts und Tabellen in Funktionsaufrufen.

7.50. Using

Funktion

Durch `Using` können Werte an InputScript Parameter übergeben werden. Sowohl Direktwerte als auch Feldwerte können übergeben werden. In dem aufgerufenen InputScript muss durch `Parameter` ein gleichnamiger Parameter angelegt sein.

Beispiele

```
Pushbutton (toolbar) "Anlegen" "CRE1" Process="AnlegenDirekt.txt"
   Using Waers = "DEM"
   Using Betrag = [Betrag2]

On "Enter"  Process="AnlegenDirekt.txt"
   Using Waers = "DEM"
   Using Betrag = [Betrag2]

On "/11" Fcode="/0"  Process="AnlegenDirekt.txt"
   Using Waers = "DEM"
   Using Betrag = [Betrag2]

Hier erhält Waers den Wert "DEM", Betrag den Wert des Feldes
F[Betrag2].
```

Format

```
Using pname = "Wert"

Using pname = [feldname]
```

Tips &Tricks

- Bitte beachten Sie den Unterschied bei folgenden Notationen:

  A: `Using CC = "&F[Buchungskreis]"`

  B: `Using CC = [Buchungskreis]`

  C: `Using CC = "Buchungskreis"`

  D: `Using CC = &F[Buchungskreis]`

  Falls das Feld `F[Buchungskreis]` zunächst den Wert `"0001"` enthielt und dann vom Benutzer mit `"5800"` überschrieben wurde, erhält der Parameter `CC` folgende Werte:

  A: `CC = "0001"`

  B: `CC = "5800"`

  C: `CC = "Buchungskreis"`

  D: `CC = "0001"` (gleichbedeutend mit Variante A)

  Im allgemeinen ist nur Variante B sinnvoll

8. GuiXT Controls

8.1. Funktionsübersicht

- Einbetten von ActiveX Controls in SAP GUI
- Anzeige eines ActiveX Objekts im SAP GUI Fenster
- Aufruf von VBScript Routinen
- Zugriff auf GuiXT Variablen aus VBScript
- RFC Call aus VBScript
- Behandlung von 'events' des ActiveX Controls wie z.B. Klick auf eine Zeile
- Einbinden des Internet Explorer Contols mit der in ConnectHTML beschriebenen Schnittstelle

8.2. Installation / Konfiguration

Die Komponente GuiXT Controls wird gleichzeitig mit dem GuiXT Setup installiert. Folgen Sie hierzu bitte der entsprechenden Anleitung.

Die Standard SAP GUI Installation enthält nicht die GuiXT Controls Komponente (Datei guixtcontrols.dll). Es genügt dann, die Datei guixtcontrols.dll in das SAP GUI Verzeichnis (meist C:\Programme\SAP\FrontEnd\SAPgui) zu kopieren. Andernfalls erhalten Sie im GuiXT Protokoll Fenster eine entsprechende Fehlermeldung " guixtcontrols.dll not found".

Wenn Sie GuiXT installiert haben, benötigen Sie zum Einsatz der GuiXT Controls einen Lizenzschlüssel. Diesen können Sie mit dem entsprechenden Formular anfordern und anschließend im GuiXT Profile eintragen.

Die VBScript Dateien werden im Profil unter "VBScript library" angegeben. Sie können dort mehrere Dateien nennen und diese auch wie die GuiXT Scripte im SAP Web Repository ablegen.

© by Synactive GmbH

9. Die Befehle im Detail (GuiXT Controls)

9.1. Control

Funktion	Mit der `Control`-Anweisung können Sie ein ActiveX Control in ein SAP GUI Fenster einbetten.
Beispiel	`Control (10.5,40.2) (20.5,80) progid="Comctl.ListViewCtrl" name="cid"` Das ListView Control aus den "Microsoft Common Controls" wird in dem Screen eingeblendet. Falls die Scriptvariable V[cid] bereits eine Referenz auf ein gültiges ListView Control enthält, wird dieses angezeigt. Ansonsten wird ein neues ActiveX Objekt angelegt und die Referenz darauf in der Scriptvariable V[cid] zurückgeliefert.
Format	`Control (Zeile1,Spalte1) (Zeile2,Spalte2) progid="ProgId" name="vname"` Die Angaben `(Zeile,Spalte1)` und `(Zeile2,Spalte2)` stellen die linke obere Ecke und die rechte untere Ecke des Anzeigebereichs dar. Es können Dezimalzahlen (mit Punkt, nicht Komma) angegeben werden, um das Bild exakt zu positionieren. Folgende Formate sind bei `progid=` möglich: • Angabe der ProgID der ActiveX Komponente. Beispiele: `progid=`"Comctl.ListViewCtrl" List View Control aus den Microsoft Common Controls `progid=`"Comctl.TreeCtrl" Tree View Control aus den Microsoft Common Controls `progid=`"Shell.Explorer" Internet Explorer Control `progid=`"myclass.control1" Eine selbst implementierte ActiveX Komponente (z.B. in VB.NET programmiert) • Angabe der technischen CLSID ("Class Identification". Beispiel `progid=`"{CA8A9780-280D-11CF-A24D-444553540000}" • Angabe einer URL. In diesem Fall wird das Explorer Control mit der angegebenen URL geöffnet. Beispiele: `progid=`"http://www.google.de" `progid=`"file:///c:/demo/d1.html" • Direkte Angabe von HTML-Code in der Notation "MSHTML:...". In diesem Fall wird das Explorer Control mit dem angegebenen HTML-Code geöffnet. Beispiel: `progid=`"MSHTML: Hello "

	Die Kommunikation mit dem Control erfolgt durch VBScript Routinen, siehe CallVBS. Es ist auch möglich, auf Ereignisse des controls zu reagieren "event handling", siehe Objekt "guixt" in VBScript. Durch if Q[ok] können Sie abfragen, ob das Control erzeugt werden konnte.
Zusatzoptionen	**-floating** — Zur Anzeige wird automatisch ein neues Fenster geöffnet, das unabhängig vom SAP GUI Fenster verschoben und in der Größe verändert werden kann. **-maximize** — Das Fenster wird maximiert angezeigt. Meist in Verbindung mit -floating **initflag==** — Angabe eines Variablennamens. In die Variable wird "X" gestellt, wenn das Control neu erzeugt wurde, sonst "". Man damit steuern, ob eine VBScript Initialisierungsroutine aufgerufen werden soll.
Tipps & Tricks	• Die auf dem PC installierten ActiveX Controls werden im GuiXT Script Editor über Tools->Display ActiveX Controls angezeigt • Die von einer ActiveX Komponente zur Verfügung gestellten Attribute und Methoden können in VBScript Editoren wie z.B. VbsEdit angezeigt werden.

9.2. CallVbs

Funktion	Mit der **CallVBS**-Anweisung können Sie eine VBScript-Funktion aufrufen. Voraussetzung: Komponente "GuiXT Controls" ist aktiv.
Beispiel	`CallVBS msg = test1` Die VBScript-Funktion "test1" wird aufgerufen. Das Ergebnis der Funktion wird als String in die Variable V[msg] gestellt.
Format 1	`CallVBS funcname "par1" "par2 "par2" ...` Die VBScript Funktion "funcname" wird aufgerufen. Die angegebenen Strings "par1", "par2",... werden an die Funktion als Parameter übergeben (Wertübergabe). Über die Notation "&V[varname]" als Parameter können Sie dabei den Inhalt von Scriptvariablen übergeben.
Format 2	`CallVBS varname = funcname "par1" "par2 "par2"` Die VBScript Funktion "funcname" wird aufgerufen. Die angegebenen Strings "par1", "par2",... werden an die Funktion als Parameter übergeben (Wertübergabe). Das Resultat der Funktion wird in einen String umgewandelt und in die Variable V[varname]

	gestellt.
Sicherheit	VBScript ist in ähnlicher Weise wie z.B. Visual Basic, Java oder C++ in der Lage, auf einem PC Schaden anzurichten, zum Beispiel Dateien zu löschen oder Registry-Werte zu ändern. Deshalb sind vor der Verteilung von VBScript-Dateien im Prinzip die gleichen Anforderungen an Test und Qualitätskontrolle zu stellen wie an direkt ausführbare Programme anderer Programmiersprachen.

Details zum Ablauf	Bei der ersten Ausführung einer **CallVBS**-Anweisung pro SAP GUI Modus werden alle im GuiXT Profile angegebenen VBScript Bibliotheken (.vbs Dateien) geladen. Die Funktionsnamen und globalen Variablen müssen über alle angegebenen Dateien hinweg eindeutig sein, da beim Laden der Dateien nur ein einziger globaler Namensraum verwendet wird. Anweisungen in den VBScript-Dateien, die sich ausserhalb von Funktionen befinden, werden beim Laden sofort ausgeführt. Globale VBScript Variablen behalten innerhalb eines SAP GUI Modus ihren Wert.
	
	Sie können die VBScript-Dateien im SAP Web Repository ablegen und zentral pflegen, was die Qualitätskontrolle erleichtert. In der Konfigurationsdatei "guixt.ini" werden durch die folgenden Anweisungen drei VBScript Dateien aus dem SAP Web Repository geladen:
	
	`VBSLibrary SAPWR:ZGUIXT.vbslib01.vbs`
	`VBSLibrary SAPWR:ZGUIXT.vbslib02.vbs`
	`VBSLibrary SAPWR:ZGUIXT.vbslib03.vbs`
	
	Syntaxfehler in einer der VBScript Bibliotheksdateien sollte man vermeiden, da das Laden der Funktionen dann mit einer Fehlermeldung abgebrochen wird. Am besten die VBScript Funktionen in einem speziellen VBScript Editor schreiben und dort die Syntax überprüfen, in dem man die Datei testweise ausführt.
	
	Die Parameterübergabe an die VBScript Funktion erfolgt als Wertübergabe von Strings, ebenso wird der Funktionswert der VBScript Funktion bei der Rückkehr als String in die GuiXT-Variable gestellt. Einzige Ausnahme: siehe den nachfolgenden Abschnitt über Objektreferenzen.
	
	In der VBScript-Funktion können Sie direkt, lesend und schreibend, auf GuiXT Variablen und Langtextvariablen zugreifen, siehe Objekt "guixt" in VBScript. Damit können Sie eine Referenzübergabe einer Variablen dadurch realisieren, dass Sie statt eines Wertes den Namen der Variablen an die VBScript-Funktion übergeben und dort mit guixt.get() und guixt.set() darauf zugreifen.

Objektreferenzen	VBScript ist eine objektbasierte Sprache, in der Sie Objekte dynamisch erzeugen können. Dabei kann es sich entweder um ActiveX Komponenten handeln, die Sie mit **CreateObject()** anlegen oder über die GuiXT Control-Anweisung erzeugen, oder um Objekte zu selbst definierten VBScript-Klassen (**Class ... End Class**), die Sie in VBScript mit **New** erzeugen. Die Parameterübergabe von Objektreferenzen an GuiXT Variablen ist in beide Richtungen möglich. GuiXT benutzt dazu eine String-Darstellung der Objektreferenz

und konvertiert automatisch zwischen der Stringdarstellung und der Objektreferenz (technisch: eine "IDispatch" Schnittstelle). Sie können also zum Beispiel in VBScript eine Klasse definieren, in einer VBScript Funktion ein Objekt der Klasse mit New anlegen und die Objektreferenz an GuiXT zurückgeben. An eine weitere VBScript-Funktion können Sie dann aus GuiXT die Objektreferenz als Parameter übergeben und in der Funktion auf Attribute und Methoden des Objekts zugreifen.

Da VBScript eine untypisierte Sprache ist, sollte man die Komplexität der VBScript Funktionen nicht zu gross werden lassen. Bei Bedarf können Sie eigene ActiveX Komponenten in einer typisierten Sprache wie Visual Basic .NET oder C# implementieren und diese in VBScript verwenden.

Debugging

Sie können mit einer "Stop" Anweisung in der VBScript Funktion den Debugger aktivieren. Falls Sie nicht ein spezielles Tools wie "Microsoft Script Debugger" oder "Visual Studio" installiert haben, wird der im Internet Explorer integrierte Script Debugger aufgerufen.

Bitte beachten:

Falls Sie im Internet Explorer -> Extras -> Internet Optionen -> Erweitert das Script-Debugging ausgeschaltet haben, hält der "Stop" Befehl nicht an, sondern bricht die Verarbeitung ohne Fehlermeldung ab.

Beispiel 2

Wir starten Excel mit einigen Demowerten:

```
// GuiXT Script

// set some demo values

SetV[region.1]"Americas"
SetV[sales.1]"2584"

SetV[region.2]"Europe"
SetV[sales.2]"1228"

SetV[region.3]"Asia"
SetV[sales.3]"1628"

// call excel

CallVBSstart_excel 3

' VB Script
Function start_excel(rows)

    Dim XL
    Set XL = CreateObject("Excel.Application")

    XL.Visible = True

    XL.Workbooks.Add

    XL.Columns(1).ColumnWidth = 30
    XL.Columns(2).ColumnWidth = 20

    For k = 1 to rows
```

```
        XL.Cells(k,1).Value = guixt.get("region." & k)
        XL.Cells(k,2).Value = guixt.get("sales."  & k)
    Next

End Function
```

9.3. CallVBSAsync

Funktion	Mit der **CallVBSAsync**-Anweisung können Sie eine VBScript-Funktion asyncron aufrufen. Voraussetzung: Komponente "GuiXT Controls" ist aktiv.
Beispiel	**CallVBSAsync SetGridLayout** Die VBScript-Funktion "SetGridLayout" wird asynchron aufgerufen, sobald alle Elemente des SAP GUI Fensters aufgebaut sind.
Formate und Details	Die Anweisung ist in Syntax und Semantik identisch mit **CallVBS**; der Aufruf geschieht aber nicht synchron, sondern erst nach Verarbeiten des GuiXT Scripts und nach dem Aufbau des gesamten Screens durch SAP GUI. Nur in asynchron aufgerufenen VBScript Funktionen können Sie auf die Objekte und Methoden des SAP GUI Scripting zugreifen, da diese noch nicht aufgebaut sind, wenn das GuiXT Script - das ja das Screen Layout ändern kann - abläuft. Falls das SAP GUI Window ein "SAP Tree Control" oder ein "SAP Grid Control" enthält, können Sie dieses in der asynchron aufgerufenen Funktion als Objekt "SapTree" bzw. "SapGrid" sofort verwenden und mit den Methoden des SAP GUI Scripting bearbeiten. Das "Session" Objekt von SAP GUI Scripting ist durch "SapSession" ansprechbar. Alle Objekte sind nur bekannt, wenn SAP GUI Scripting auf dem Server und dem Frontend aktiviert sind.

9.4. Object GuiXt in VBS Script

Funktion	Wenn Sie mit **CallVBS** eine VBScript-Funktion aufrufen, ist bereits ein Objekt "guixt" vordefiniert. Es gestattet den Zugriff auf GuiXT Scriptvariablen und bietet einige Sonderfunktionen z.B. RFC-Aufrufe aus VBScript. Voraussetzung: Komponente "GuiXT Controls" ist aktiv.
Get()	**r = guixt.Get(var)** Wenn var den Inhalt "xxx" hat, wird die GuiXT Scriptvariable V[xxx] gelesen und der Inhalt nach r gestellt. Beispiel: `Dim menu_icon_path`

```
menu_icon_path = guixt.Get("menu_icon_path")
```

Set()

guixt.Set(var,value)

Wenn var den Inhalt "xxx" hat, wird die GuiXT Scriptvariable V[xxx] auf den Inhalt von value gesetzt.

Beispiel:

```
guixt.Set("menu_icon_path", "C:\guixt\menu\icons")
```

GetText()

r = guixt.GetText(var)

Wenn var den Inhalt "xxx" hat, wird die GuiXT Langtextvariable text[xxx] gelesen und der Inhalt nach r gestellt.

Beispiel:

```
' read menu
Dim menu
menu = Split(guixt.GetText("menu"), vbCrLf)
```

SetText()

guixt.SetText(var,value)

Wenn var den Inhalt "xxx" hat, wird die GuiXT Langtextvariable text[xxx] auf den Inhalt von value gesetzt.

Beispiel:

```
Call guixt.settext("menu_layout",Join(s,vbCrLf))
```

Value()

r = guixt.Value(var)

GuiXT ersetzt im angegebenen String alle &-Ausdrücke nach GuiXT-Logik durch die entsprechenden Werte und stellt das Resultat nach r.

Beispiel:

```
Dim mat
mat = guixt.Value("&F[Material]")
```

Input()

guixt.Input(s)

Der angegebene String s wird exakt so interpretiert wie für die input= Option der Image-Anweisung beschrieben.

Beispiel:

```
guixt.input("U[VBELN]:" & ListView.selectedItem)
guixt.input("OK:/nVA03,process=va03_enter.txt")
```

FilePath()

guixt.FilePath(s, [cache=True])

Der angegebene String s wird als Dateiname relativ zum GuiXT Scriptverzeichnis interpretiert.

- Der gesamte Dateiname, mit aktuellem Scriptverzeichnis, wird zurückgegeben.
- Ein absoluter Dateiname wie C:\guixt\scripts\..." bleibt unverändert.
- Bei Angabe einer Datei aus dem SAP Web Repository "SAPWR:..." wird im GuiXT Cache eine lokale Kopie erstellt und der Cache-Dateiname zurückgegeben
- Ein Angabe einer http oder ftp Datei wie "http://..." oder "ftp://" wird im GuiXT Cache eine lokale Kopie erstellt und der Cache-Dateiname zurückgegeben.

Bei cache=**True** werden externe Dateien zunächst im GuiXT Cache gesucht.

Beispiele:

```
imgpath = guixt.input("images\home.gif")
imgpath ist jetzt z.B. "C:\guixt\scripts1\images\home.gif"

imgpath =
guixt.input("http://www.synactive.com/images/s10logo_blue.gif")
imgpath ist jetzt z.B.
"C:\Users\ADMINI~1\AppData\Local\Temp\guixt\GuiXT
Cache\http\www.synactive.com\images\s10logo_blue.gif"
```

SetEventHandler()

guixt.SetEventHandler(obj1,obj2)

Angabe zweier Objekte. Für jedes Event z.B. "Click" von Objekt obj1 wird die Funktion "OnClick" von Objekt2 aufgerufen. Die Parameterübergabe erfolgt gemäss der Event-Schnittstelle, d.h. Sie müssen die Funktionen zur Eventbehandlung mit den entsprechenden Parametern anlegen. Meist ist obj1 ein Control, z.B. ein Tree View oder ein List View, und obj2 ist ein mit VBScript -Mitteln definiertes Objekt zur Behandlung der Events.

Tipp 1: Die ausgelösten Events und aktuellen Parameter können Sie mit der Trace-Funktion im GuiXT-Window anzeigen.

Tipp 2: Die möglichen Events und ihre Parameter werden in VBScript Editoren wie z.B. VbsEdit dargestellt.

Beispiel (Details dazu siehe Tutorial 4 zu GuiXT Controls):

```
' handle  events
Set e = New va03_list_events
Set e.ListView = lv
Call guixt.SetEventHandler(lv, e)

mit

Class va03_list_events

    Dim ListView
```

```
    Function OnColumnClick(ColumnHeader)

      ...

    End Function

End Class
```

DoEvents()

guixt.DoEvents()

Die Funktion wartet einige Millisekunden und verarbeitet dann alle Windows-Nachrichten, die sich in der Warteschlange befinden.

Sie benötigen die Funktion bei einigen Controls, die asynchrone Verarbeitungen durchführen. Zum Beispiel werden im Internet Explorer eine Seite und die einzelnen Objekte asynchron geladen, und Sie müssen die eigene Verarbeitung durch DoEvents() unterbrechen, bis die Seite geladen ist, ehe Sie mit der Verarbeitung der geladenen Seite beginnen können.

Beispiel:

```
ie.Navigate(url)

' wait until document is loaded
Const READYSTATE_LOADING = 1

Do
  guixt.DoEvents()
Loop While ie.ReadyState = READYSTATE_LOADING
```

CreateObject()

guixt.CreateObject(progid)

Die Funktion ist weitgehend identisch mit der VBScript-Funktion CreateObject(). Der Unterschied besteht darin, dass die GuiXT Controls-Funktion CreateObject() eine statisch vordefinierte Version der Windows Common Controls "Comctl. ..." benutzt.

Beispiel:

```
Set imgList = guixt.CreateObject("ComCtl.ImageListCtrl")
```

Rfc()

guixt.Rfc(arguments,...)

Aufruf eines SAP Funktionsbausteins über RFC. Die Parameter sind identisch mit den bei Call beschriebenen Parametern, wobei nicht ein gesamter String, sondern die einzelnen Teile als Parameter übergeben werden.

Die GuiXT Variable V[_exception] enthält nach dem Rfc() Aufruf entweder "", falls kein Fehler auftrat, oder einen Fehlertext. Sie können auf die Variable mit guixt.Get("_exception") zugreifen.

Für übergebene Tabellen wird ein VBScript-Array aus Strings verwendet, d.h. jede

Tabellenzeile ist ein String des Arrays. Falls Sie die Tabelle nur für die Ausgabe des Funktionsbausteins benötigen, reicht die Definition mit "Dim"; es wird dann automatisch ein Array in der benötigten Grösse angelegt.

Beispiel:

```
Dim all_vkorg

' read data via RFC / SAP searchhelp
Call guixt.Rfc("Z_S10_SEARCHHELP", "in.SEARCHHELP", "H_TVKO",
"in.COLUMNS",
                "VKORG(4),VTEXT(20)",  "table.DATA(WIDTH:24)",
all_vkorg)

For Each row In  all_vkorg

   Dim vkey
   Dim vtext

   vkey  = Mid(row,1,4)
   vtext = Mid(row,5,20)

   ...

Next
```

10. Viewer

10.1. Funktionsübersicht

- Anzeige von html, rtf, bmp, jpg, gif Dateien in R/3 Bildern.
- Interaktion zwischen html und rtf Dateien mit R/3
    - Setzen von Werten in die R/3 Eingabefelder bei Mausklick in html-Anzeige
    - Auslösen von Aktionen im R/3 bei Mausklick in html-Anzeige
    - Anzeigetransaktionen in html Format mit automatischer Übernahme der R/3-Werte
- Eingabe von Werten in html mit Verarbeitung durch JavaSkript und Weitergabe an R/3

10.2. Installation / Konfiguration

Der Viewer wird gleichzeitig mit dem GuiXT Setup installiert. Folgen Sie hierzu bitte der entsprechenden Anleitung. Die Datei pinnacle.dll muß in ein geeignetes WIndows Verzeichnis (z.B. C:\WINNT\System32), oder in das SapGui Verzeichnis (e.g. C:\Programs\SAPPC\sapgui)kopiert werden. Andernfalls erhalten Sie im GuiXT Protokoll Fenster eine Fehlermeldung "pinnacle.dll not found".

Wenn Sie GuiXT installiert haben, benötigen Sie zum Einsatz des Viewer lediglich einen Lizenzschlüssel. Diesen können Sie mit dem entsprechenden Formular anfordern und anschließend im GuiXT Profile eintragen.

10.3. ViewHelp

Funktion	Eine Variante der **View** Anweisung finden Sie in der Option **ViewHelp**, die in Kombination mit den Schlüsselwörtern **Mark** und **Image** zur Verfügung steht. Mit der **ViewHelp** Option können Sie eine html Datei bei Bedarf über ein Hilfefenster einblenden, z.B. um während einer Schulung Hinweise zur SAP Nutzung bereitzustellen oder wenn Sie detaillierte Infos zu einem Produkt anbieten möchten. Bei Mausklick auf das Hilfesymbol in der Toolbar oder bei einzelnen Feldern erscheint die hinterlegte Dokumentation (html-Format) am rechten Bildschirmrand. Die Größe des Anzeigefensters läßt sich mit der Maus verändern. In die **ViewHelp** Datei können Sie sowohl Werte einstreuen, die bei Mausklick übernommen werden, als auch Funktionen oder neue Transaktionen direkt abrufbar anzeigen. Sobald Sie auf ein SAP Feld klicken, verschwindet die Anzeige der Hilfedatei automatisch. Vergleichen Sie hierzu das Viewer Tutorial 7 Hilfstexte und Grafiken einblenden mit ViewHelp.
• MARK	mit Option ViewHelp

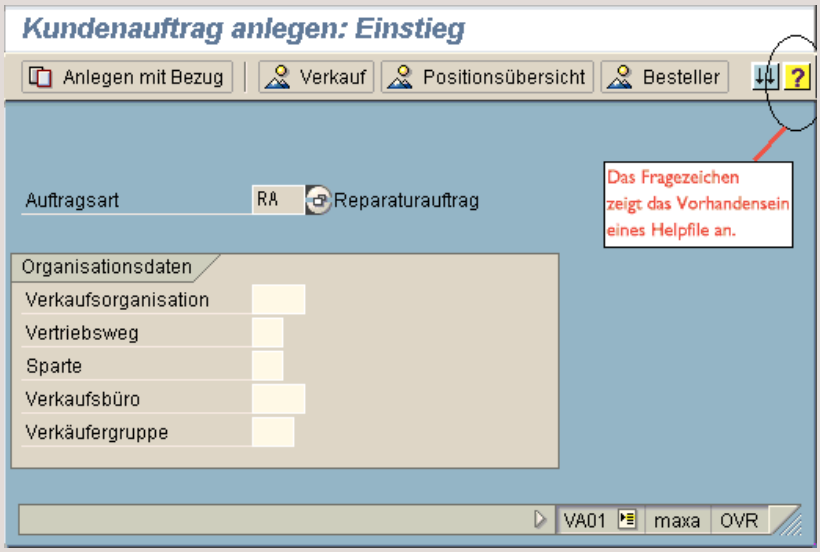
Anzeige eigener Hilfedateien in html-Format mit der `Mark` Anweisung	Durch einen Mausklick auf das Markierungskreuz neben dem Feldnamen wird im rechten oberen Teil des SAP-Fensters die unter **ViewHelp=** angegebene html-Datei zur Anzeige gebracht. Sie kann Interaktionen mit SAP enthalten, siehe **View**-Befehl (Setzen von Feldinhalten, Aufruf von Funktionen). Sobald der Benutzer auf ein SAP Feld klickt, verschwindet die Anzeige der Hilfedatei automatisch. `Mark [Eingabefeld] ViewHelp="filename.html"`
Beispiel	`Mark [Gewichtseinheit] Image="help.gif" ViewHelp="doc_gwe.htm"`
Optionen für `ViewHelp`	**ViewHelpWidth=** Breite der Help-Anzeige (in Pixel). Falls keine Breite angegeben ist, wird 40% der Fensterbreite verwendet. **ViewHelpHeight=** Höhe der Help-Anzeige (in Pixel). Falls keine Höhe angegeben ist, wird die gesamte Fensterhöhe verwendet. **ViewHelpPosition=** Position (linke obere Ecke) der Help-Anzeige. Angabe in der Form (Zeile,Spalte), z.B. ViewHelpPosition=(10,50), oder auch relativ zu Bildelementen, z.B. ViewHelpPosition="[Material]+(1,30)". Falls eine Position, aber nicht Breite und Höhe angegeben sind, wird das gesamte Fenster ab der angegebenen Position (nach rechts/unten) zur Anzeige verwendet.
Beispiel	`Mark [Gewichtseinheit] Image="help.gif" ViewHelp="doc_gwe.htm" ViewHelpWidth="200"`

- IMAGE mit Option ViewHelp

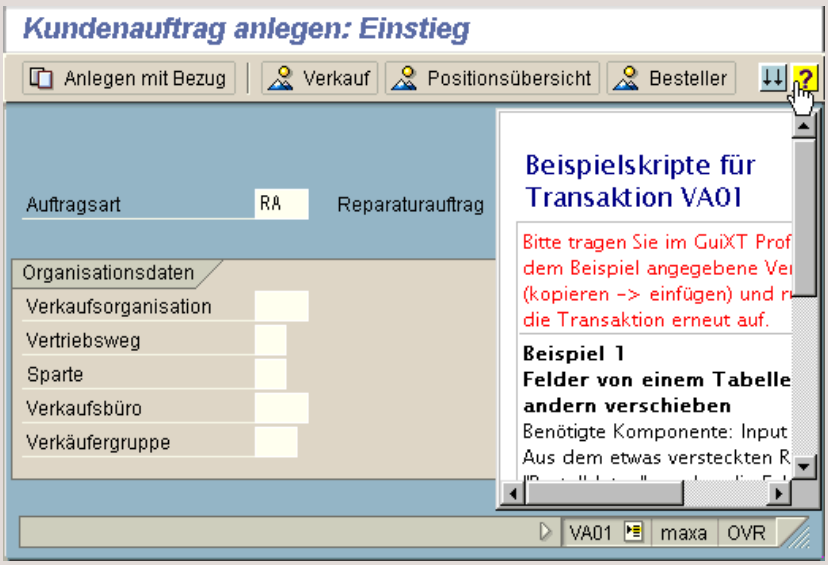
Anzeige eigener Hilfedateien in html-Format mit der `Image` Anweisung	*Hierfür wird die Komponente "Viewer" benötigt* Durch einen Mausklick auf das Bild wird im rechten oberen Teil des SAP-Fensters die unter **ViewHelp=** angegebene html-Datei zur Anzeige gebracht. Sie kann Interaktionen mit SAP enthalten, siehe **View**-Befehl (Setzen von Feldinhalten, Aufruf von Funktionen). Sobald der Benutzer auf ein SAP Feld klickt, verschwindet die Anzeige der Hilfedatei automatisch. `Image (zeile,spalte) "bilddatei" ViewHelp="filename.html"`
Beispiel	`Image (toolbar) "help.gif" ViewHelp="doc_meldung.htm"`

Optionen für `ViewHelp`	`ViewHelpWidth=`	Breite der Help-Anzeige (in Pixel). Falls keine Breite angegeben ist, wird 40% der Fensterbreite verwendet.
	`ViewHelpHeight=`	Höhe der Help-Anzeige (in Pixel). Falls keine Höhe angegeben ist, wird die gesamte Fensterhöhe verwendet.
	`ViewHelpPosition=`	Position (linke obere Ecke) der Help-Anzeige. Angabe in der Form (Zeile,Spalte), z.B. ViewHelpPosition=(10,50), oder auch relativ zu Bildelementen, z.B. ViewHelpPosition="[Material]+(1,30)". Falls eine Position, aber nicht Breite und Höhe angegeben sind, wird das gesamte Fenster ab der angegebenen Position (nach rechts/unten) zur Anzeige verwendet.
Beispiel	`Image (toolbar) "help.gif" ViewHelp="doc_meldung.htm" ViewHelpWidth="200"`	

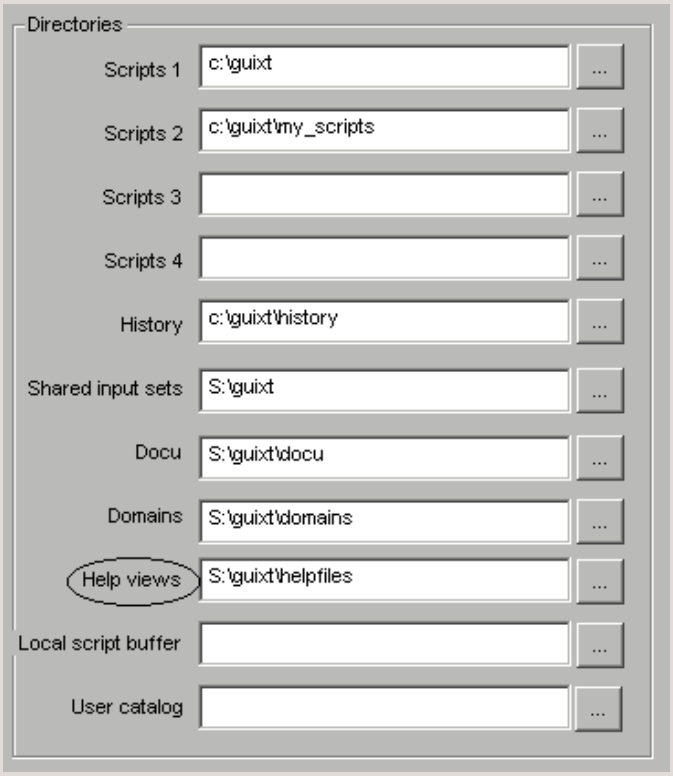
10.4. Helpview Verzeichnisse



Im GuiXT Profile können Sie ein eigenes Verzeichnis einrichten, in dem generell die HelpView-Dateien zu den einzelnen SAP Transaktionen abgelegt werden. Sie brauchen dann keine eigene Anweisung in einem GuiXT Skript zu schreiben, um die HelpView-Datei einzubinden, sondern GuiXT zeigt automatisch das Vorhandensein einer HelpView-Datei in der Toolbar mit einem frei wählbaren Symbol an.



Die HelpView-Datei im html Format (in diesem Fall **d.transaction.VA01.html**) erscheint bei Mausklick am rechten Bildrand. Die Größe des Anzeigefensters läßt sich mit der Maus verändern.



Help views Verzeichnis mit eigenen html-Hilfedateien (Viewer). Pro Transaktion kann in dem angegebenen Verzeichnis eine Datei abgelegt werden, z.B. d.transaction.VA01.html für Transaktion VA01, Sprache D.
Zusätzlich muss das Verzeichnis eine Datei help.gif enthalten (Format 18x18 Pixel, z.B. ?), die von GuiXT als Ikone rechts in der toolbar angezeigt wird, falls zu der aktuellen Transaktion eine entsprechende Hilfedatei existiert. Durch eine eigene Image-Anweisung mit ViewHelp=... kann die angegebene Datei im GuiXT Skript übersteuert werden.
Im SAP logon-Bild wird die Datei d.transaction.logon.html gesucht, wobei der Sprachenschlüssel gemäss der Sprache des logon-Bildes eingesetzt wird.

Das Help Views Verzeichnis kann auf dem PC, im lokalen Netzwerk oder auf einem zentralen ftp oder http Server eingerichtet werden. Vgl. GuiXT Profile.

Eintrag in der guixt.ini Datei:

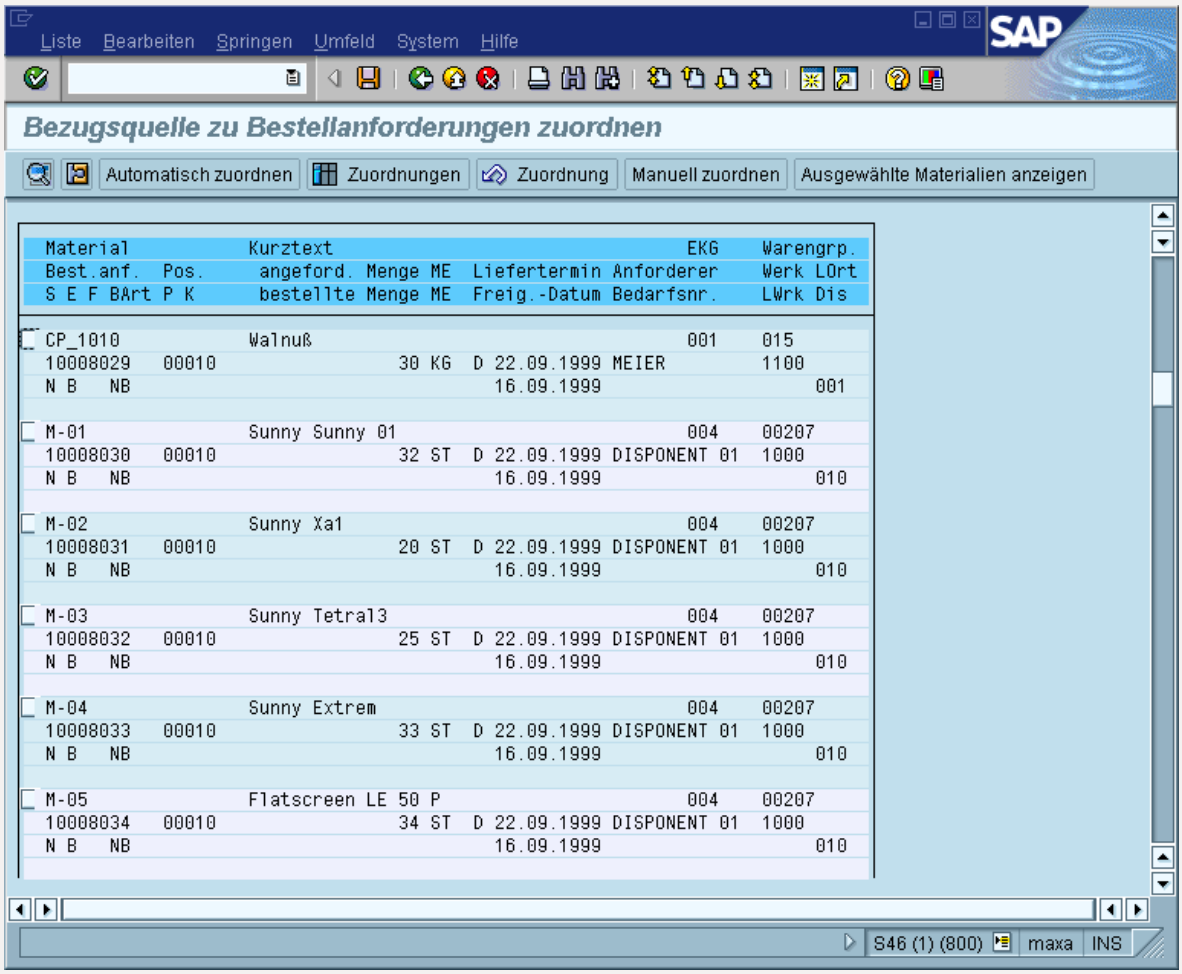
```
HelpViewDirectory
```

© by Synactive GmbH

11. Special Topics

11.1. Daten aus Listen lesen

InputScripts müssen manchmal Daten lesen, die in SAP in Listenform dargestellt werden. Ein typisches Beispiel ist die folgende Liste, die angezeigt wird, wenn Transaktion ME56 "Bestellanforderungen zuordnen" aufgerufen wird:


Liste Bearbeiten Springen Umfeld System Hilfe
SAP
Bezugsquelle zu Bestellanforderungen zuordnen
Automatisch zuordnen
Zuordnungen
Zuordnung
Manuell zuordnen
Ausgewählte Materialien anzeigen
Material Kurztext EKG Warengrp.
Best.anf. Pos. angeford. Menge ME Liefertermin Anforderer Werk LOrt
S E F BArt P K bestellte Menge ME Freig.-Datum Bedarfsnr. LWrk Dis
CP_1010 Walnuß 001 015
10008029 00010 30 KG D 22.09.1999 MEIER 1100
N B NB 16.09.1999 001
M-01 Sunny Sunny 01 004 00207
10008030 00010 32 ST D 22.09.1999 DISPONENT 01 1000
N B NB 16.09.1999 010
M-02 Sunny Xa1 004 00207
10008031 00010 20 ST D 22.09.1999 DISPONENT 01 1000
N B NB 16.09.1999 010
M-03 Sunny Tetra13 004 00207
10008032 00010 25 ST D 22.09.1999 DISPONENT 01 1000
N B NB 16.09.1999 010
M-04 Sunny Extrem 004 00207
10008033 00010 33 ST D 22.09.1999 DISPONENT 01 1000
N B NB 16.09.1999 010
M-05 Flatscreen LE 50 P 004 00207
10008034 00010 34 ST D 22.09.1999 DISPONENT 01 1000
N B NB 16.09.1999 010
S46 (1) (800)
maxa
INS

Das "screen elements" Fenster von GuiXT ist in diesem Fall völlig leer und die Bildnummer ist immer 0120.

Bitte beachten Sie: Es ist nicht möglich, Listen mit GuiXT zu verändern,
aber Sie können Daten aus Listen in InputScripts übernehmen und Sie können weitere Möglichkeiten für Benutzer mit den GuiXT Skripten hinzufügen.

Die folgenden Systemvariablen stehen für Listen zur Verfügung:

- _listline(1), _listline(2),...
  Zeilen, die auf der aktuellen Seite angezeigt werden.
  _listline(1) ist die erste Zeile nach dem Seitentitel

- _listline

Aktuelle Zeile (abhängig von der Cursorposition)

- _liststring
  Aktueller Textstring (abhängig von der Cursorposition, Begrenzung nach links und rechts ist ein Leerzeichen)
- _listcursorrow, _listcursorcol
  Spalten- und Zeilen-Nummer der Cursorpostiton
- _listlastrow
  Nummer der letzten Zeile der Liste (auf allen Seiten, ohne die Zeile des Seitentitels)
- _listfirstvisiblerow, _listlastvisiblerow
  Erste und letzte Zeile, die auf der aktuellen Seite sichtbar ist

Automatisches Skrollen

- Enter "/ScrollToLine=..."
  Skrollt/Rollt die Liste bis zu Zeile ...

Beispiel 1 (basiert auf SAP Rel. 4.6C)

Sie fügen einen Pushbutton "Ausgewähltes Material anzeigen" über der ME56 Liste ein. Wenn der Benutzer eine Materialnummer auswählt (cursor)
und anschließend auf den Pushbutton klickt, dann wird die Anzeigetransaktion MM03 "Material anzeigen" in einem neuen Modus aufgerufen,
die ausgewählte Materialnummer wird übergeben und mit Enter bestätigt.

GuiXT Script RM06BL00.D0120.txt

```
Pushbutton (toolbar) "Ausgewähltes Material anzeigen" "/OMM03" "Process=Materialanzeigen.txt"
using MATNR = [_liststring]
```

InputScript Materialanzeigen.txt

```
Parameter MATNR

Screen SAPLMGMM.0060
  Set F[Material] &[MATNR]
  Enter
```

Beispiel 2 (basiert auf SAP Rel. 4.6C)

Sie möchten eine html Datei anzeigen, die alle Bestellanforderungen enthält, mit zusätzlichen Angaben wie Preis, Wert, Herkunft ....

Bitte beachten Sie, dass dieses Beispiel für kleine Listen gut geeignet ist, bei größeren Datenmengen aber zu Performance Verlusten führen kann

GuiXT Script RM06BL00.D0120.txt

```
Pushbutton (toolbar) "Übersicht im html Format" "Process=ME56Uebersicht.txt"
```

InputScript ME56Uebersicht.txt

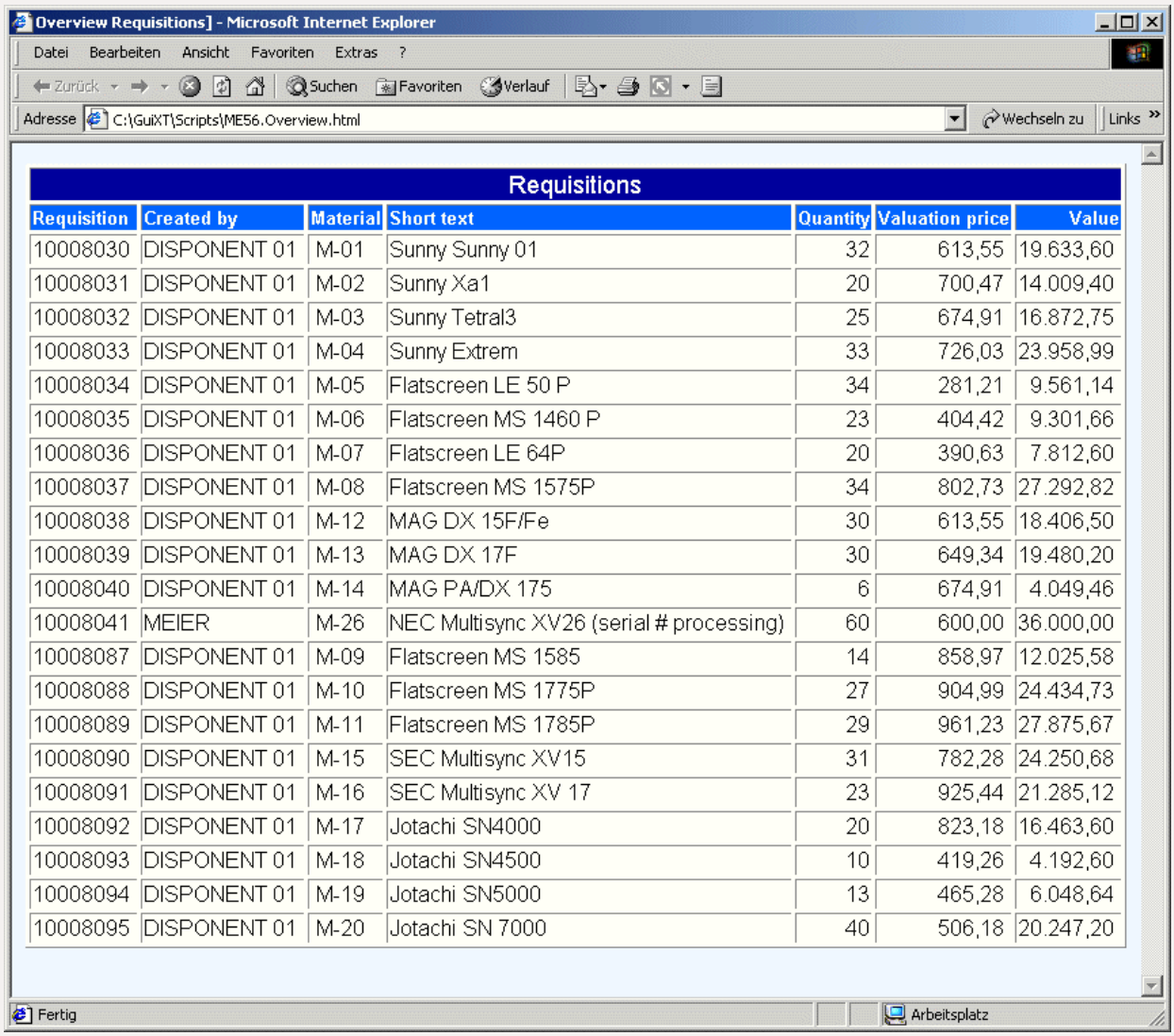
```
Screen RM06BL00.0120
  Set V[ME56.Uebersicht*] "" // Variablen zurücksetzen
  Set V[ME56.Uebersicht.index] 1
  Set V[ME56.Uebersicht.row] 1

  Enter "/ScrollToLine=1"

label next_req

Screen RM06BL00.0120
  if V[ME56.Uebersicht.row>&[_listlastrow]]
    goto end
  endif

  SetCursor (5,1) // Auf eine Zeile zeigen
  Enter "/2" // Details ansehen

Screen SAPMM06B.0102

  Set V[ME56.Uebersicht.&[ME56.Uebersicht.index].Bestellanforderung] "&[Bestellanforderung]"
  Set V[ME56.Uebersicht.&[ME56.Uebersicht.index].material] "&[Material]"
  Set V[ME56.Uebersicht.&[ME56.Uebersicht.index].text] "&[Kurztext]"
  Set V[ME56.Uebersicht.&[ME56.Uebersicht.index].Preis] "&[BwrtPreis]"
  Set V[ME56.Uebersicht.&[ME56.Uebersicht.index].Menge] "&[Menge]"

  Enter "=AS"

Screen SAPMM06B.0110

  Set V[ME56.Uebersicht.&[ME56.Uebersicht.index].Gesamtwert] "&[Gesamtwert]"
  Set V[ME56.Uebersicht.&[ME56.Uebersicht.index].Disponent] "&[Disponent]"

  Enter "/3" // Return

Screen SAPMM06B.0102
  Enter "/3" // Return

Screen RM06BL00.0120
  Set V[ME56.Uebersicht.row] &V[ME56.Uebersicht.row] + 4
  Set V[ME56.Uebersicht.index] &V[ME56.Uebersicht.index] + 1
  Enter "/ScrollToLine=&[ME56.Uebersicht.row]"
  goto next_req
```

```
label end

  View "ME56Uebersicht.html" "template=ME56Uebersichttemplate.html"
  Enter "/ScrollToLine=1"
```

html Seite ME56Uebersichttemplate.html:

```
<html>

<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<meta http-equiv="Content-Language" content="de">
<meta name="ProgId" content="FrontPage.Editor.Document">
```

Ergebnis der html Anzeige:



Requisitions

Requisition	Created by	Material	Short text	Quantity	Valuation price	Value
10008030	DISPONENT 01	M-01	Sunny Sunny 01	32	613,55	19.633,60
10008031	DISPONENT 01	M-02	Sunny Xa1	20	700,47	14.009,40
10008032	DISPONENT 01	M-03	Sunny Tetral3	25	674,91	16.872,75
10008033	DISPONENT 01	M-04	Sunny Extrem	33	726,03	23.958,99
10008034	DISPONENT 01	M-05	Flatscreen LE 50 P	34	281,21	9.561,14
10008035	DISPONENT 01	M-06	Flatscreen MS 1460 P	23	404,42	9.301,66
10008036	DISPONENT 01	M-07	Flatscreen LE 64P	20	390,63	7.812,60
10008037	DISPONENT 01	M-08	Flatscreen MS 1575P	34	802,73	27.292,82
10008038	DISPONENT 01	M-12	MAG DX 15F/Fe	30	613,55	18.406,50
10008039	DISPONENT 01	M-13	MAG DX 17F	30	649,34	19.480,20
10008040	DISPONENT 01	M-14	MAG PA/DX 175	6	674,91	4.049,46
10008041	MEIER	M-26	NEC Multisync XV26 (serial # processing)	60	600,00	36.000,00
10008087	DISPONENT 01	M-09	Flatscreen MS 1585	14	858,97	12.025,58
10008088	DISPONENT 01	M-10	Flatscreen MS 1775P	27	904,99	24.434,73
10008089	DISPONENT 01	M-11	Flatscreen MS 1785P	29	961,23	27.875,67
10008090	DISPONENT 01	M-15	SEC Multisync XV15	31	782,28	24.250,68
10008091	DISPONENT 01	M-16	SEC Multisync XV 17	23	925,44	21.285,12
10008092	DISPONENT 01	M-17	Jotachi SN4000	20	823,18	16.463,60
10008093	DISPONENT 01	M-18	Jotachi SN4500	10	419,26	4.192,60
10008094	DISPONENT 01	M-19	Jotachi SN5000	13	465,28	6.048,64
10008095	DISPONENT 01	M-20	Jotachi SN 7000	40	506,18	20.247,20

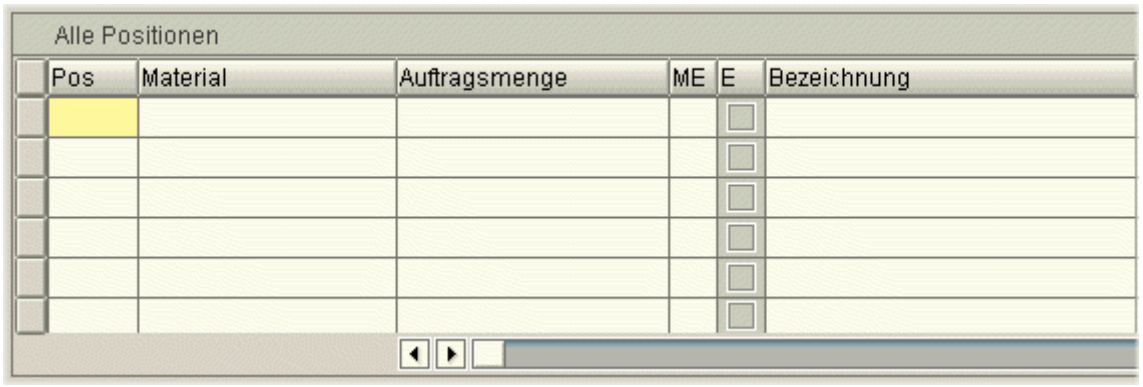
11.2. Tabellen in GuiXT und InputAssistant

Einige R/3 Bildschirme enthalten Daten in Tabellenform, die sog. table controls.

R/3 Standardbildschirm (VA01 Auftragserfassung, drittes Bild).

Mit GuiXT können Sie das Layout dieser Tabellen anpassen; Sie können auch Zellenwerte einlesen, Zellenwerte bestimmen und Spalten auswählen.

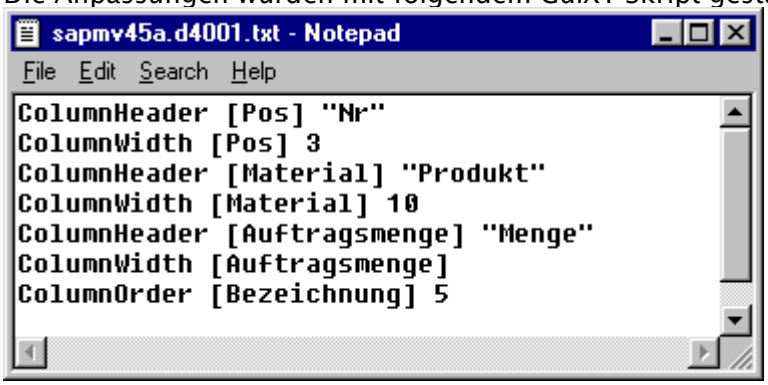
Spaltenüberschriften können ebenso leicht angepaßt werden wie Spaltenbreite und Spaltenreihenfolge. Sie können für Ihr Unternehmen überflüssige Spalten ausblenden, selten verwendete Spalten ans Ende setzen, Spaltenüberschriften an den Sprachgebrauch Ihres Unternehmens anpassen. Im folgenden Beispiel wird das dritte Bild der Transaktion `VA01` verwendet.



R/3 Standardbildschirm (VA01 Auftragserfassung, drittes Bild, Detail).

Die gleiche Tabelle, die mit mit GuiXT angepasst wurde.

Die Anpassungen wurden mit folgendem GuiXT Skript gestaltet:



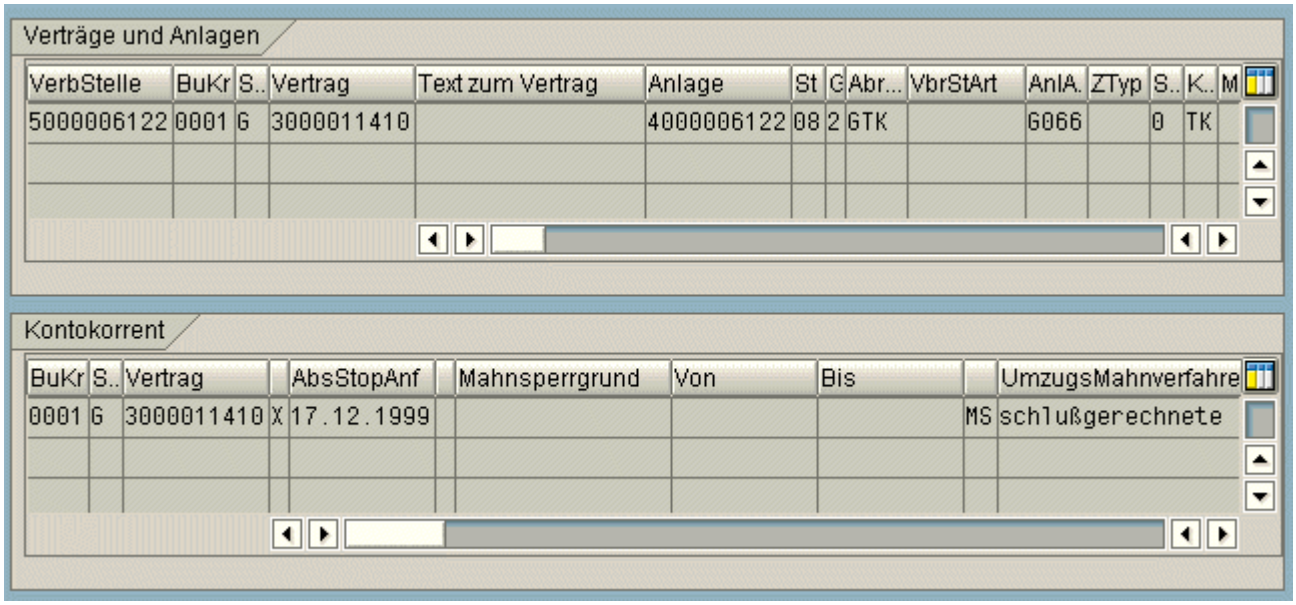
GuiXT Skript im Editor

Format der Tabellenanweisungen

Wenn die Tabelle einen Titel *title* hat, z.B. "Alle Positionen" dann ist der Name der Tabelle **[*title*]** oder **T[*title*]**, z.B. `[Alle Positionen]` oder `T[Alle Positionen].`

Für Tabellen ohne Titel ist der Name `[table]` oder `T[table]`. Wenn es zwei oder mehr Tabellen ohne Titelangaben gibt, dann wird die erste Tabelle mit `[table]` oder `T[table]` bezeichnet und die folgende Tabelle bekommt `[table.2]` eine Ziffer angehängt, beginnend mit 2: `[table.2]` oder **T[table.2]**

Beachten Sie, dass der Rahmen um die Tabelle einen eigenen Titel haben kann, der anders lautet als der Tabellentitel. Oft hat auch die Tabelle in einem Rahmen keinen Titel, wie im folgenden Beispiel.


Verträge und Anlagen
VerbStelle BuKr S.. Vertrag Text zum Vertrag Anlage St G Abr... VbrStArt AnlA. ZTyp S.. K.. M
5000006122 0001 G 3000011410 4000006122 08 2 GTK 6066 0 TK
Kontokorrent
BuKr S.. Vertrag AbsStopAnf Mahnsperrgrund Von Bis UmzugsMahnverfahre
0001 G 3000011410 X 17.12.1999 MS schlußgerechnete

Hier haben nur die Rahmen die Titel "Verträge und Anlagen" bzw. "Kontokorrent", die Tabellen haben keinen Namen. GuiXT gebraucht in diesem Fall (titellose Tabellen) auch die Namen [table] und [table.2]. Wenn Tabellentitel und Rahmentitel nicht leicht zu unterscheiden sind, klicken Sie bitte auf **View→ Screen Elements** im GuiXT Fenster, um die richtigen Tabellennamen herauszufinden.

Übersicht über die GuiXT Anweisungen, die für Tabellen wichtig sind. (Siehe auch die Dokumentation der entsprechenden Anweisungen)

- `pos` verschiebt die ganze Tabelle zu einer neuen Position.
  ```
  pos [Alle Positionen] (10,24)
  ```

- `del` löscht die ganze Tabelle.
  ```
  del [Alle Positionen]
  ```

- `NoInput` verhindert die Eingabe für alle Zellen der Tabelle.
  ```
  NoInput [Alle Positionen]
  ```

- `ColumnHeader` verändert eine Spaltenübersicht.
  ```
  ColumnHeader [Alle Positionen,Wrk] "Werk"
  ```

- `ColumnWidth` ändert die Spaltenbreite.
  ```
  ColumnWidth [Alle Positionen,Beschreibung] 36
  ```

- `ColumnOrder` ändert die Spaltenreihenfolge.
  ```
  ColumnOrder [Alle Positionen,Wrk] 6
  ColumnOrder [Alle Positionen,Datum] 7
  ```

- `&cell[...]` ist eine Variable, die durch den aktuellen Zellwert ersetzt wird. Wird vor allem verwendet, um Werte einer Maske zu speichern, während Sie ein InputScript ausführen.
  ```
  Set V[product1] "&cell[Alle Positionen,Material,1]"
  Set V[product2] "&cell[Alle Positionen,Material,2]"
  ```
  Notation: `&cell[Tabellenname,Spaltenname,Zeile]`
  Der Spaltenname kann auch eine Zahl 1,2,3, ... sein, die sich auf die Originalspaltenzahl bezieht. Die Spaltenzahl beginnt mit 1.

- `Set cell[...]` fügt einen Wert in eine Zelle ein. Wird vor allem in InputScript für die automatische Maskenverarbeitung verwendet.
  ```
  Set cell[Alle Positionen,Material,1] "&[Produkt1]"
  Set cell[Alle Positionen,Material,2] "&[Produkt1]"
  ```
  Notation: `Set cell[Tabellenname,Spalte,Zeile] "Wert"`
  Der Spaltenname kann auch eine Zahl 1,2,3, ... sein, die sich auf die Originalspaltenzahl bezieht. Die Spaltenzahl beginnt mit 1.

- Besonderheit Column=0
  `Set cell[Tabellenname,0,Zeile,] "Wert"` wählt eine Spalte (Wert "X") oder deselektiert eine Spalte (Wert " ").
  Beispiel:
  ```
  Set cell[Alle Positionen,0,1] "X"
  ```

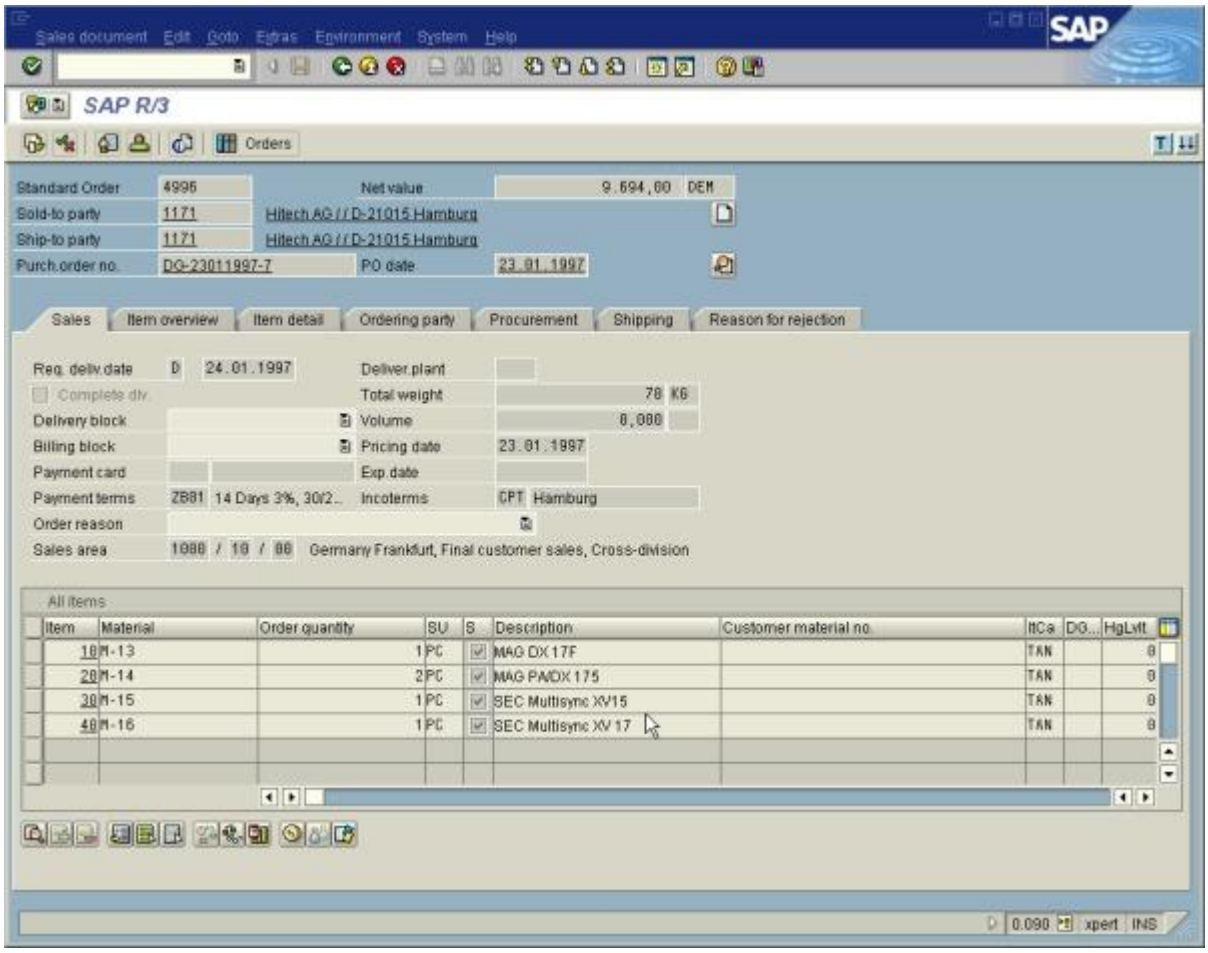
11.3. Tabellen im Aufruf von RFC-Funktionsbausteinen (call)

Funktionsaufrufe aus einem InputScript heraus sind eine schnelle Möglichkeit, zusätzliche Daten aus dem SAP System zu erhalten. Sie können hierfür sowohl Einzelfelder als auch Tabellen verwenden. Bei dynamischen Daten (wie beispielsweise Texten oder einer größeren Anzahl von Objekten) ist die Verwendung von Tabellen die beste Art, Informationen zwischen SAP System und InputScript zu transportieren.

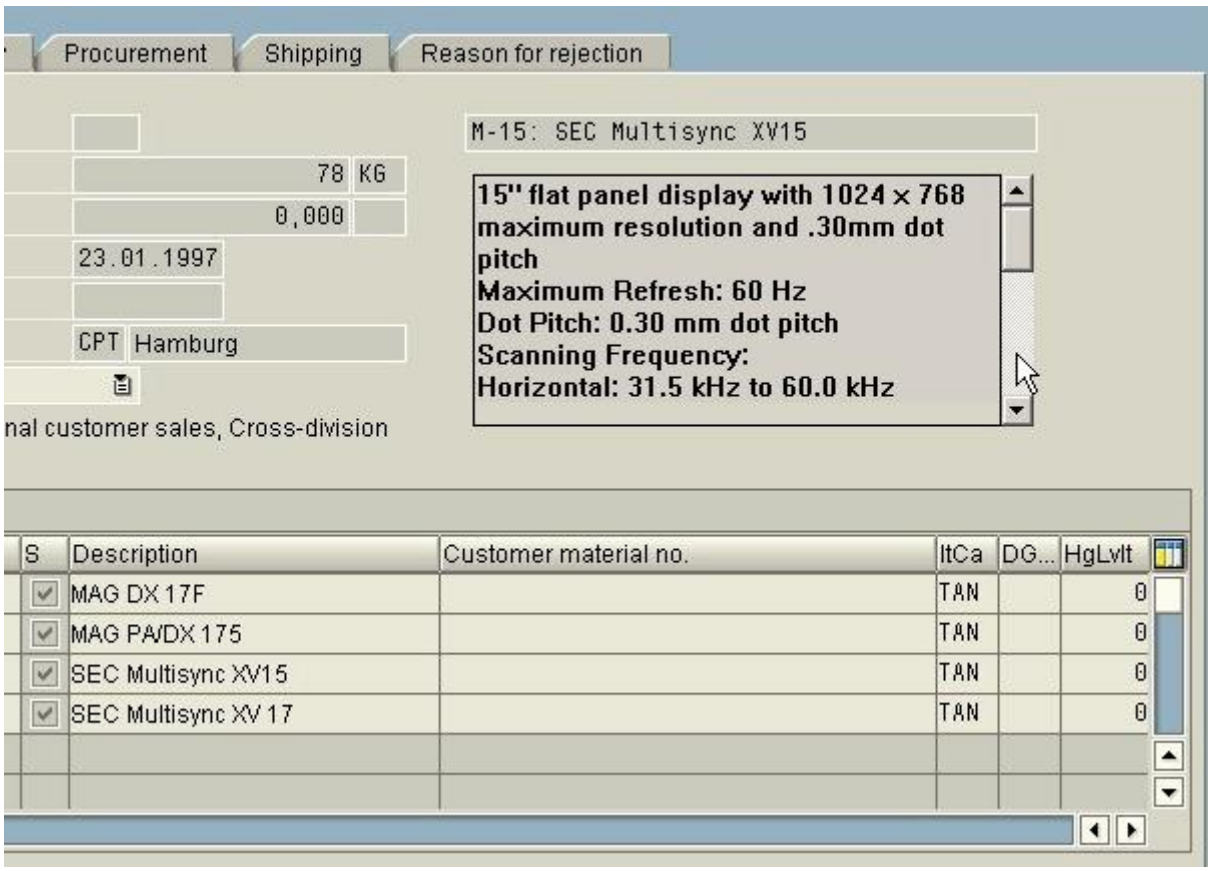
Hier stellen wir Ihnen zwei Beispiele vor. Beide lassen sich gut als Grundlage für Ihre eigenen Entwicklungen verwenden. Im zweiten Beispiel demonstrieren wir auch die Erstellung "dynamischer" Bildelemente.

Beispiel 1: Anzeige der Materialbeschreibung in VA03

In der Transaktion VA03 (Kundenauftrag anzeigen) möchten wir dem Anwender die Möglichkeit geben, durch Doppelklick auf die Materialbezeichnung in der Tabelle die ausführliche Materialbeschreibung anzuzeigen.


Sales document Edit Goto Extras Environment System Help
SAP
SAP R/3
Orders
Standard Order 4996 Net value 9.694,00 DEM
Sold-to party 1171 Hitech AG / / D-21015 Hamburg
Ship-to party 1171 Hitech AG / / D-21015 Hamburg
Purch.order no. DG-23011997-7 PO date 23.01.1997
Sales Item overview Item detail Ordering party Procurement Shipping Reason for rejection
Req. deliv.date 0 24.01.1997 Deliver.plant
Complete dlv. Total weight 78 KG
Delivery block Volume 0,000
Billing block Pricing date 23.01.1997
Payment card Exp.date
Payment terms ZB01 14 Days 3%, 30/2.. Incoterms CPT Hamburg
Order reason
Sales area 1000 / 10 / 00 Germany Frankfurt, Final customer sales, Cross-division
All items
Item Material Order quantity SU S Description Customer material no. ItCa DG... HgLvlt
10 M-13 1 PC MAG DX 17F TAN 0
20 M-14 2 PC MAG PADX 175 TAN 0
30 M-15 1 PC SEC Multisync XV15 TAN 0
40 M-16 1 PC SEC Multisync XV 17 TAN 0
0.000 xpert INS

VA03: Doppelklick auf den Materialkurztext (Tabellenspalte "Bezeichnung")



Die ausführliche Materialbeschreibung wird angezeigt.

Die folgende Technik wird angewandt:

- Im GuiXT Skript behandeln wir den Doppelklick durch eine **On "/2"** Anweisung. Daraufhin wird das InputScript VA03_doubleclick.txt ausgeführt.
  Beachten Sie den Parameter Fcode="/2" in der On Anweisung. Dies bedeutet, dass der Doppelklick an die SAP Anwendung weitergegeben wird, wenn nicht das InputScript einen anderen FUnktionscode durch "Enter" setzt.

- Im InputScript stellen wir zuerst fest, ob der Cursor auf die Spalte "Bezeichnung" zeigt, indem wir die Systemvariablen V[_tabrow] und V[_tabcol] verwenden.
  Zeigt er nicht auf diese Spalte, dann geschieht nichts, so dass der Standard Doppelklick durch die SAP Anwendung durchgeführt wird. Zeigt der Cursor auf die Spalte, dann nehmen wir die Materialnummer und die Kurzbezeichnung aus dieser Zeile. Wir lösen "Enter" aus, um die Standard Doppelklick Funktion in diesem Fall auszuschalten.

- Das GuiXT Skript wird nach diesem Enter erneut ausgeführt. Wir rufen eine Funktion auf, die die komplette Materialbeschreibung liefert. Diese zeigen wir in einer Textbox mit der Materialbezeichnung an. Zum Schluß setzen wir die Variable für die Materialnummer zurück, so dass die Textbox beim nächsten Enter wieder verschwindet.

GuiXT script "SAPMV45A.E4001.TXT"

```
if Q[Transaction=VA03] and Q[Page=Verkauf]

  // Bei Doppelklick
  On "/2" Fcode="/2" Process="VA03_doubleclick.txt"
```

```
  // Text anzeigen? (nach Doppelklick)
  if V[VA03_DisplayTextMatnr]
    Call "ZZ_GUIXT_MATERIAL_TEXT" in.artnr="&V[VA03_DisplayTextMatnr]"
table.text="matxt"

    Text (7,84) "&V[VA03_DisplayTextMatnr]: &V[VA03_DisplayTextDesc]" size=40 -
border
    TextBox (8,84) (14,124) name="matxt" -readonly

    Set V[VA03_DisplayTextMatnr] ""
  endif
endif
```

InputScript "VA03_doubleclick.txt"

```
// Cursor in Spalte "Bezeichnung"   (intern nr 6)?
if V[_tabrow>0] and V[_tabcol=6]

  Set V[VA03_DisplayTextMatnr] "&cell[Alle Positionen,Material,&V[_tabrow]]"
  Set V[VA03_DisplayTextDesc]  "&cell[Alle Positionen,6,&V[_tabrow]]"

  // Ersetze "/2" mit normalem "Enter"
  Enter

endif
```

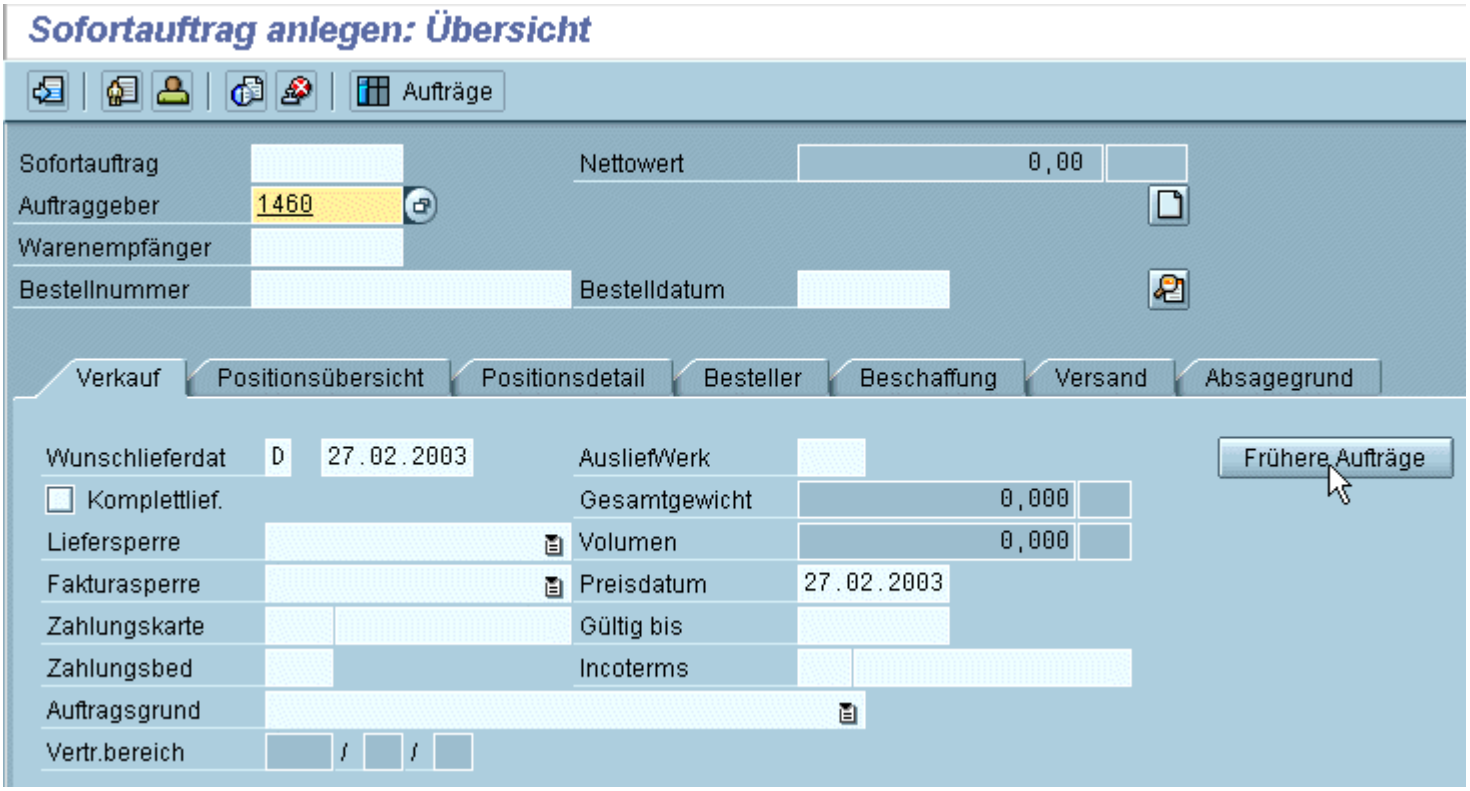
Function "ZZ_GUIXT_MATERIAL_TEXT"

```
FUNCTION ZZ_GUIXT_MATERIAL_TEXT.
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*" IMPORTING
*" VALUE(ARTNR) TYPE ARTNR
*" VALUE(LANGUAGE) LIKE SY-LANGU DEFAULT SY-LANGU
*" VALUE(ID) LIKE THEAD-TDID DEFAULT 'GRUN'
*" TABLES
*" TEXT STRUCTURE W3HTML OPTIONAL
*"----------------------------------------------------------------------


DATA: TNAME LIKE THEAD-TDNAME.
  TNAME = ARTNR.

data: lt_stream type w3_html occurs 10.

Refresh text.

DATA: LT_SSTEXT LIKE TLINE OCCURS 0.

CALL FUNCTION 'READ_TEXT'
   EXPORTING ID       = ID
             LANGUAGE = LANGUAGE
             NAME     = TNAME
             OBJECT   = 'MATERIAL'
   TABLES
             LINES    = LT_SSTEXT
   EXCEPTIONS
            OTHERS = 1.

IF SY-SUBRC EQ 0.
```

```
  CALL FUNCTION 'CONVERT_ITF_TO_STREAM_TEXT'
       TABLES
           ITF_TEXT    = LT_SSTEXT
           TEXT_STREAM = lt_stream.

  LOOP AT lt_stream INTO text.
    append text.
  ENDLOOP.
ENDIF.

ENDFUNCTION.
```

Beispiel 2: Produkte vorschlagen bei "Kundenauftrag anlegen" (VA01)

In der Transaktion VA01 (Kundenauftrag anlegen) gibt der Anwender eine Nummer für den Auftraggeber ein. Für diesen Auftraggeber schlagen wird dann einige Produkte aus früheren Aufträgen vor. Wir zeigen diese Produkte als Pushbuttons an, und ein einfacher Mausklick auf einen Pushbutton fügt dann automatisch die Materialnummer und Auftragsmenge in den neuen Auftrag ein. Gleichzeitig zeigen wir in 5 Pushbuttons der letzten 5 Aufträge mit dem Bestelldatum an. Ein Mausklick auf diesen Pushbutton springt zur Anzeige des jeweiligen Auftrags.



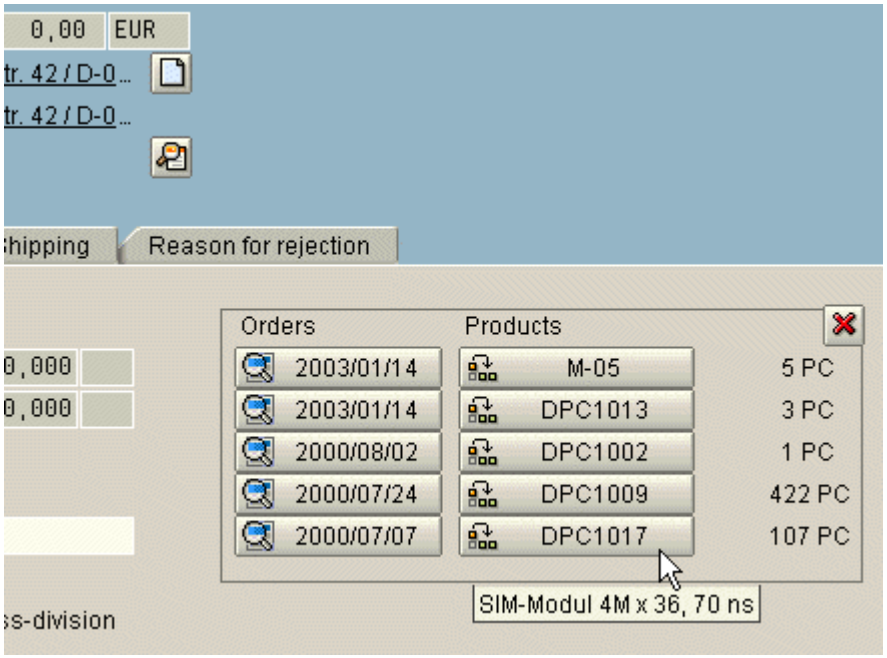
VA01: Neuer Pushbutton "Frühere Aufträge"

Nach Eingabe der Kundennummer für den Auftraggeber kann der Anwender auf den neuen Pushbutton klicken, um folgende Informationen zu erhalten:

- **5 frühere Aufträge**
- **5 Produkte, die aus diesen Aufträgen stammen, mit Auftragsmenge und Kurzbezeichnung (als**

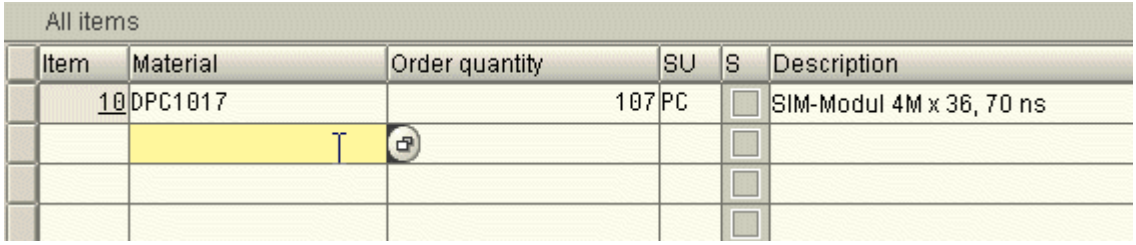
Tipp bei jedem Pushbutton)

Wenn Sie mehr als 5 Produkte anzeigen möchten, dann können Sie einen oder mehrere Pushbuttons hinzufügen. Wenn Sie eine größere Produktanzahl zeigen möchten, dann ist es empfehlenswert, ein HTML Template mit dem Add-on Viewer zu benutzen. Die Technik, die wir Ihnen hier vorstellen, bleibt im Wesentlichen die gleiche, mit dem Unterschied, dass die Produkt-Buttons mit einer html Tabelle gezeigt werden.



Die zusätzlichen Produktinformationen

Ein Mausklick auf den Produktbutton fügt die Materialnummer und Auftragsmenge ein; ein Mausklick auf den Auftragsbutton zeigt den Auftrag an (unter Verwendung der Transaktion VA03). Wenn das Fenster nicht mehr benötigt wird, kann der Anwender es mit dem roten Pfeil schließen.



Automatische Eingabe von Materialnummer und Auftragsmenge

Die Skripte (für ein IDES 4.6D System) sind nicht besonders lang, sie stellen allerdings höhere Ansprüche an die GuiXT Programmierung. Es wird hierbei ein dynamischer Index verwendet. Wenn Sie die Beispielskripte anpassen und verändern, dann beachten Sie bitte folgende Einschränkungen, die bei Tabellen Transfers mit der Call Anweisung gelten:

- Verwenden Sie nur zeichenartige Felder in Ihrer Tabellendefinition (keine "integer" oder "gepackten" Zahlen)
- Die maximale Standard Tabellenbreite ist 256. Verwenden Sie den width:xxx Parameter wenn Sie eine größere Tabellenbreite benötigen, z.B.. `table.ORDERS(width:500)=orders`. Hierbei

ist die maximale definierbare Breite 32000.

- Bezüglich der Zeilenanzahl gibt es keine Einschränkungen.

GuiXT Skript "SAPMV45A.E4001.TXT"

```
if Q[Transaction=VA01] and Q[Page=Verkauf]

 if V[VA01_KUNNR=&F[Auftraggeber]] and V[VA01_KUNNR]
  Offset (7,86)
  Box (0,0) (6,44)

  // close box
  Pushbutton (0,43) "@02@" Process="GetOrdersMaterials.txt"
     using KUNNR = ""

  Text (0,1) "Aufträge"

  // Index
  Set V[i] 1
  Set V[row] 1

  label next_order

  if not V[VA01_VBELN&V[i]]
    goto end_of_orders
  endif

  Pushbutton (&V[row],1) "@16\QOrder &V[VA01_VBELN&V[i]]@&V[VA01_AUDAT&V[i]]"
"/OVA03" Process="DisplayOrder.txt"
     using VBELN = "&V[VA01_VBELN&V[i]]"

  Set V[i] &V[i] + 1
  Set V[row] &V[row] + 1

  if V[i<6]
    goto next_order
  endif

  label end_of_orders

  // Index
  Set V[i] 1
  Set V[row] 1

  Text (0,17) "Produkte"


  label next_material

  if not V[VA01_MATNR&V[i]]
    goto end_of_materials
  endif

  Pushbutton (&V[row],17) "@40\Q&V[VA01_ARKTX&V[i]]@&V[VA01_MATNR&V[i]]"
process="AddMaterial.txt" size=(1,16)
    using MATERIAL = "&V[VA01_MATNR&V[i]]"
    using AMOUNT = "&V[VA01_AMOUNT&V[i]]"


  Text (&V[row],34) "&V[VA01_AMOUNT&V[i]] &V[VA01_VRKME&V[i]]"
```

```
  Set V[i] &V[i] + 1
  Set V[row] &V[row] + 1

  if V[i<6]
    goto next_material
  endif

  label end_of_materials
  else

  Pushbutton (7,86) "Frühere Aufträge" Process="GetOrdersMaterials.txt"
    using KUNNR = [Auftraggeber]
  endif
endif
```

InputScript "DisplayOrder.txt"

```
Parameter VBELN

Screen sapmv45a.0102
  Set F[Order] "&U[VBELN]"
  Enter
```

InputScript "AddMaterial.txt"

```
Parameter MATERIAL
Parameter AMOUNT

Set V[i] 1

label next_line
Set V[Material] "&cell[.,Material,&V[i]]"

// empty?
If not V[Material]

  Set cell[Alle Positionen,Material,&V[i]] "&U[MATERIAL]"
  Set cell[Alle Positionen,Auftragsmenge,&V[i]] "&U[AMOUNT]"
  Leave
endif

Set V[i] &V[i] + 1
if V[i<20]
  goto next_line
endif
```

Function "ZZ_GUIXT_CUSTOMER_ORDERS"

```
FUNCTION ZZ_GUIXT_CUSTOMER_ORDERS.
*"*"Local interface:
*" IMPORTING*" VALUE(KUNNR) TYPE KUNNR
*" VALUE(N_ORDERS) TYPE I DEFAULT 5
*" TABLES
*" ORDERS STRUCTURE ZZORDER
*" MATERIALS STRUCTURE ZZMATERIAL
*"----------------------------------------------------------------------
Tables: vakpa, vbap.


*  Activate for debugging in RFC mode:
*  call function 'SYSTEM_ATTACH_GUI'.
*  Break-point.
```

```
* add leading 000... for numerical customer numbers
if kunnr co ' 0123456789'.
  unpack kunnr to kunnr.
endif.

refresh: orders, materials.
select AUDAT VBELN from vakpa
  into corresponding fields of vakpa
      up to n_orders rows
        where kunde = kunnr and parvw = 'AG'
          order by audat descending.
  orders-audat = vakpa-audat.
  orders-vbeln = vakpa-vbeln.
  Append orders.

  select MATNR KWMENG VRKME ARKTX from vbap
   into corresponding fields of vbap
     where vbeln = orders-vbeln.

     read table materials with key vbap-matnr.
     IF sy-subrc ne 0.
       materials-matnr = vbap-matnr.
       materials-arktx = vbap-arktx.
       materials-vbeln = vakpa-vbeln.
*      Use "Write to" instead of "Move" for special fields
       Write vbap-kwmeng to materials-menge unit vbap-vrkme.
       Write vbap-vrkme to materials-vrkme.
       Append materials.
     endif.
  endselect.

endselect.
endfunction.
```

InputScript "GetOrdersMaterials.txt"

```
Parameter KUNNR

// Save customer number
Set V[VA01_KUNNR] "&U[KUNNR]"


// no customer number? Then no action
if not V[VA01_KUNNR]
  Leave
endif

// Reset order and material table
Set V[empty] ""
CopyText fromString=empty toText=orders
CopyText fromString=empty toText=materials

Call "ZZ_GUIXT_CUSTOMER_ORDERS" in.KUNNR="&U[KUNNR]" table.ORDERS=orders
table.MATERIALS=materials

// Index
Set V[i] 1

label next_order
CopyText fromText=orders toString=Pos line=&V[i]
if not Q[ok]
  // Reset Order
  Set V[VA01_ORDER&V[i]]
```

```
  goto materials
endif

Set V[VA01_AUDAT&V[i]] "&V[Pos](1-4)/&V[Pos](5-6)/&V[Pos](7-8)"
Set V[VA01_VBELN&V[i]] "&V[Pos](9-18)"

Set V[i] &V[i] + 1

if V[i<6]
  goto next_order
endif


label materials
Set V[i] 1

label next_material
CopyText fromText=materials toString=Pos line=&V[i]

if not Q[ok]
  // Reset Material
  Set V[VA01_MATNR&V[i]]
  Leave
endif


Set V[VA01_MATNR&V[i]] "&V[Pos](1-18)"
Set V[VA01_AMOUNT&V[i]] "&V[Pos](19-33)"
Set V[VA01_VRKME&V[i]] "&V[Pos](34-36)"
Set V[VA01_MATVBELN&V[i]] "&V[Pos](37-46)"
Set V[VA01_ARKTX&V[i]] "&V[Pos](47-86)"

Set V[i] &V[i] + 1

if V[i<6]
  goto next_material
endif
```

11.5. Listen in HTML-Format anzeigen (call)

Users often need to get additional information about business objects they are dealing with in a transaction. Sometimes the SAP standard system already offers these possibilities, on a button or in the menu. In other cases the only way for the user is to open an additional mode, and to navigate to the right transaction or to call up the right report, and then, in most cases, enter the object keys again.

With InputAssistant and Viewer there are various ways to implement an easy way to additional information. The technique that we want to explain here is as follows:

- You look for a standard ABAP report that displays the information you need, or you write a suitable new ABAP report
- WIth GuiXT you add a new button on the screen that invokes an InputScript, passing the necessary key information to it
- In the InputScript you call a function module that produces the wanted ABAP list in html format, and returns it to the script
- With the Viewer component you finally display the list

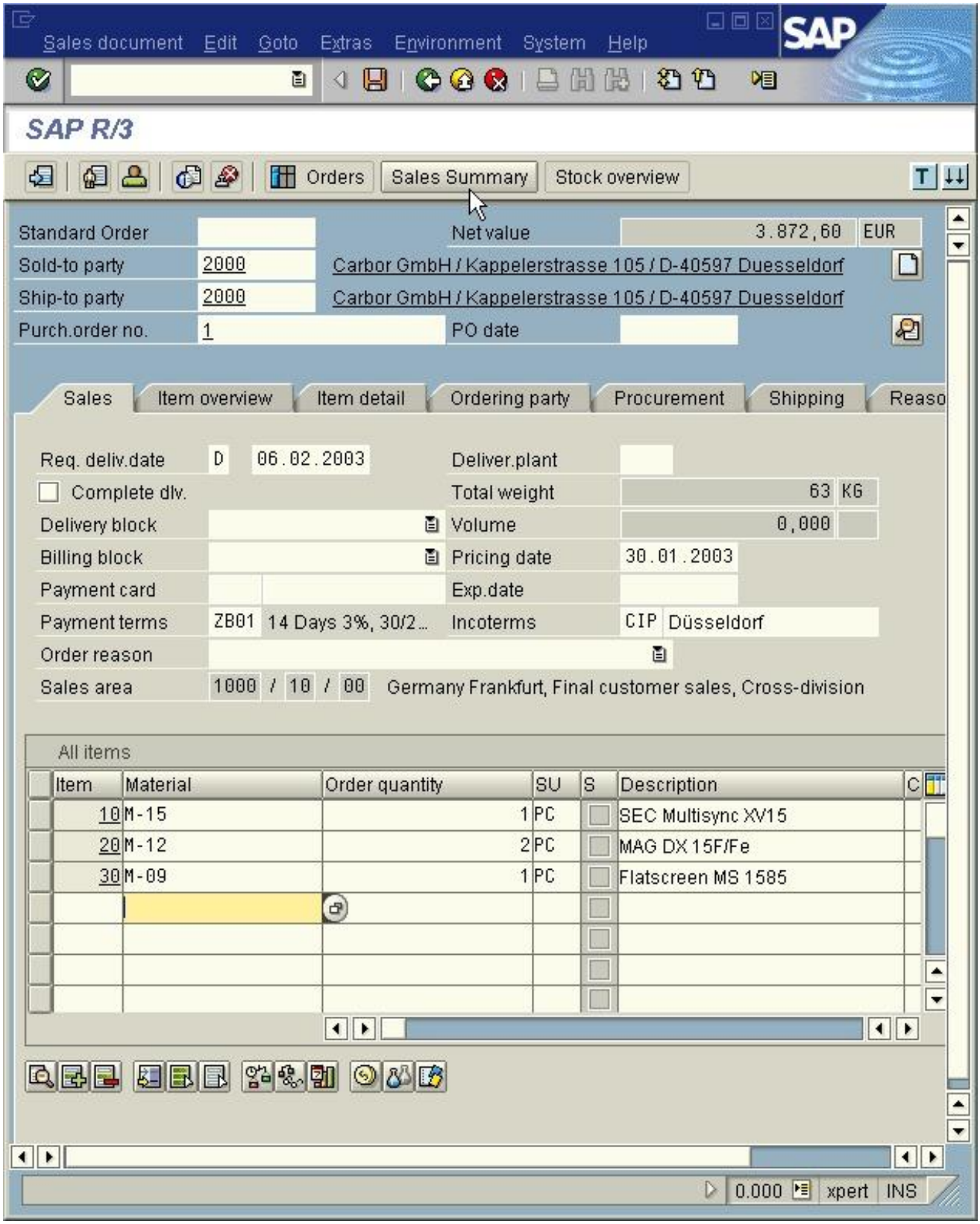
This approach has several advantages:

- Often there exists already a standard report that you can use, or at least that is a good basis for your own implementation
- Only one function call is needed, no extra roundtrips between frontend and application server
- The user can scroll the list locally, and can work with the list independently of the SAP sessions
- You can also easily combine several lists in one display

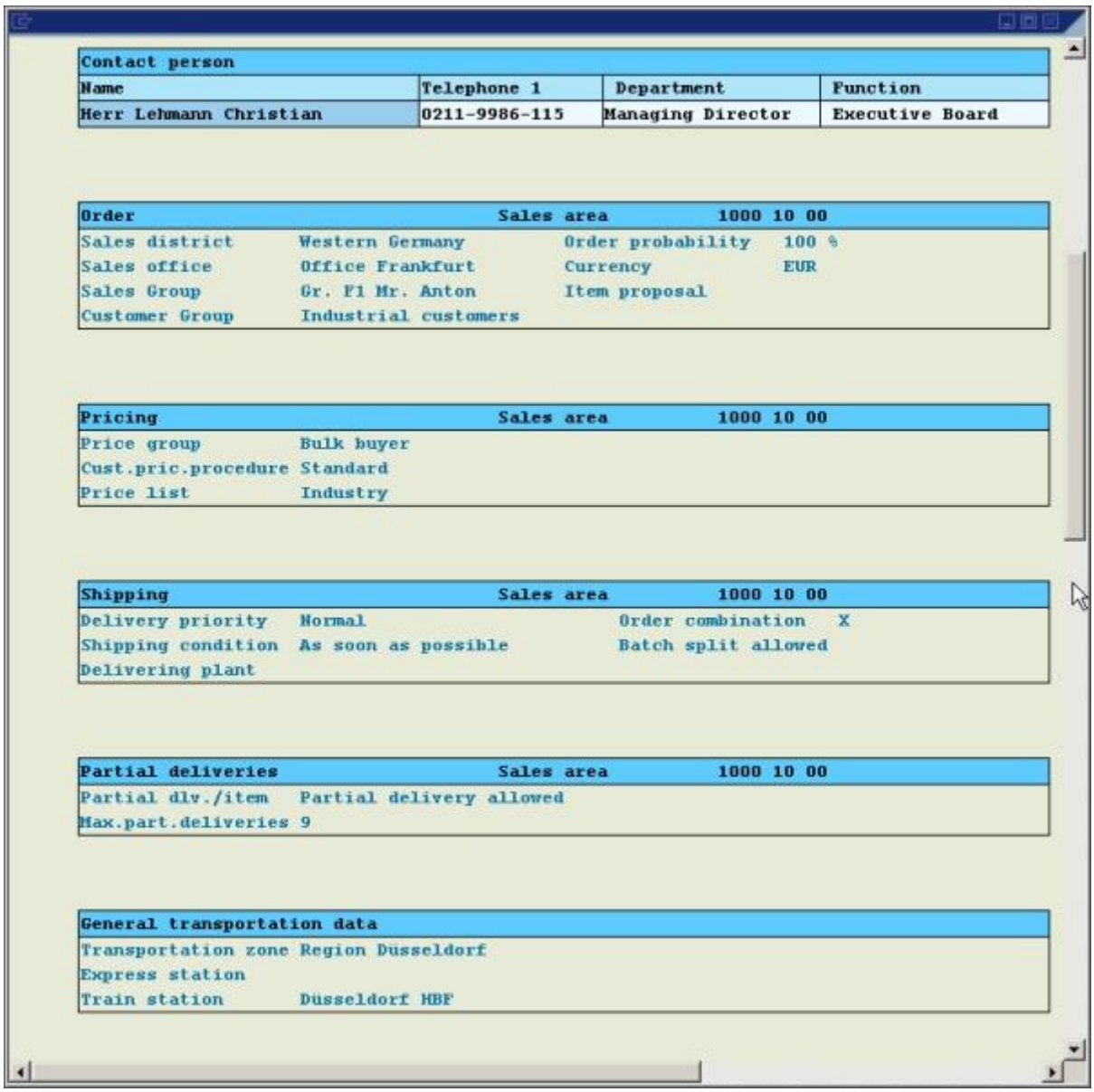
An alternative is to call the ABAP report via a /O... command (new session), and to use the normal SAP display. This is somewhat easier to implement, and requires no extra function call. On the other hand, the advantages of local scrolling, and ability of combining several lists into one, are then lost.

Example: Displaying stock overview and sales summary in VA01

In the order entry transaction VA01, we offer two new buttons in the toolbar: "Sales summary" and "Stock overview". The first button shows, for the given customer, the sales of the past, using a standard report. The second pushbutton shows the stock overview for all materials on the current table page. It also uses a standard report, produced for each material and then combined into one list.



VA01: new pushbuttons "Sales Summary and "Stock overview"



Contact person			
Name	Telephone 1	Department	Function
Herr Lehmann Christian	0211-9986-115	Managing Director	Executive Board

Order		Sales area	1000 10 00
Sales district	Western Germany	Order probability	100 %
Sales office	Office Frankfurt	Currency	EUR
Sales Group	Gr. F1 Mr. Anton	Item proposal	
Customer Group	Industrial customers		

Pricing		Sales area	1000 10 00
Price group	Bulk buyer		
Cust.pric.procedure	Standard		
Price list	Industry		

Shipping		Sales area	1000 10 00
Delivery priority	Normal	Order combination	X
Shipping condition	As soon as possible	Batch split allowed	
Delivering plant			

Partial deliveries		Sales area	1000 10 00
Partial dlv./item	Partial delivery allowed		
Max.part.deliveries	9		

General transportation data	
Transportation zone	Region Dusseldorf
Express station	
Train station	Düsseldorf HBF

Sales summary report in a separate browser window

Material M-15 SEC Multisync XV15
Material type HAWA Trading goods
Unit of measure PC Base unit of measure PC

Cl/CC/Plant/SLoc/Batch D	Unrestricted use	Qual. inspection	Reserved
Total	493,000	0,000	100,000
1000 IDES AG	243,000	0,000	100,000
1000 Werk Hamburg	29,000	0,000	0,000
0001 Materiallager	29,000	0,000	0,000
1200 Dresden	214,000	0,000	100,000
0001 Materiallager	214,000	0,000	0,000
0120 Werksverkauf	0,000	0,000	0,000
R1/M2/0120			

Material M-15 SEC Multisync XV15
Material type HAWA Trading goods
Unit of measure PC Base unit of measure PC

Cl/CC/Plant/SLoc/Batch D	Unrestricted use	Qual. inspection	Reserved
3000 IDES US INC	250,000	0,000	0,000
3000 New York	250,000	0,000	0,000
0001 Warehouse 0001	250,000	0,000	0,000

Material M-12 MAG DX 15F/Fe
Material type HAWA Trading goods
Unit of measure PC Base unit of measure PC

Cl/CC/Plant/SLoc/Batch D	Unrestricted use	Qual. inspection	Reserved
Total	571,000	0,000	100,000
1000 IDES AG	321,000	0,000	100,000
1000 Werk Hamburg	30,000	0,000	0,000
0001 Materiallager	30,000	0,000	0,000
1200 Dresden	291,000	0,000	100,000
0001 Materiallager	291,000	0,000	0,000
0120 Werksverkauf	0,000	0,000	0,000
R1/M2/0120			

Stock overview in a separate browser window

GuiXT script "SAPMV45A.E4001.TXT"

```
if Q[Transaction=VA01] and Q[Page=Sales]
  Pushbutton (toolbar) "Sales Summary"  process="SalesSummary.txt"
  Pushbutton (toolbar) "Stock overview" process="StockOverview.txt"
endif
```

InputScript "SalesSummary.txt"

```
Call "ZZ_GUIXT_CUSTOMER_PAYMENTS" in.KUNNR="&F[Sold-to party]" table.HTML=htm

// Buld temp filename using Windows TMP environment variable
Set V[VA01_viewfile] "&%[TMP]\guixtview.html"
```

```
// Write to temp file
CopyText fromText="htm" toFile="&V[VA01_viewfile]"

// Display list (coordinates: for window sizing only)
View (0,0) (30,120) "&V[VA01_viewfile]" -floating
```

InputScript "StockOverview.txt"

```
// Show stock overview for all materials on this page
Set V[material]

// del material number table
CopyText fromString="material" toText=mat

Set V[i] 1

label item
Set V[material] "&cell[All items,Material,&V[i]]"
if not V[material]
  goto end_of_page
endif
CopyText fromString="material" toText="mat" -appendline
Set V[i] &V[i] + 1
goto item

label end_of_page

Call "ZZ_GUIXT_STOCK_OVERVIEW" table.TMAT=mat table.HTML=htm

// Buld temp filename using Windows TMP environment variable
Set V[VA01_viewfile] "&%[TMP]\guixtlist.html"

// Write to temp file
CopyText fromText="htm" toFile="&V[VA01_viewfile]"

// Display list (coordinates: for window sizing only)
View (0,0) (30,120) "&V[VA01_viewfile]" -floating
```

Function "ZZ_GUIXT_CUSTOMER_PAYMENTS"

```
FUNCTION ZZ_GUIXT_CUSTOMER_PAYMENTS.
*"----------------------------------------------------------------------
*"*"Local interface:
*" IMPORTING
*" VALUE(KUNNR) TYPE KUNNR
*" TABLES
*" HTML STRUCTURE W3HTML OPTIONAL
*"----------------------------------------------------------------------

* Debugging in RFC mode:
* call function 'SYSTEM_ATTACH_GUI'.
* break-point.

data: abaplist like abaplist occurs 1 with header line.

* Add leading 000...
if kunnr co ' 0123456789'.
  unpack kunnr to kunnr.
```

```
endif.

* Produce report
Submit RVKUSTA1
    exporting list to memory
    and return
      with KUNNR EQ kunnr
      with VKORG EQ '1000'.

* get list from memory
CALL FUNCTION 'LIST_FROM_MEMORY'
     TABLES
        LISTOBJECT = abaplist.

* and convert to html
CALL FUNCTION 'WWW_HTML_FROM_LISTOBJECT'
     TABLES
        HTML = html
        LISTOBJECT = abaplist.

ENDFUNCTION.
```

Function "ZZ_GUIXT_STOCK_OVERVIEW"

```
FUNCTION ZZ_GUIXT_STOCK_OVERVIEW.
*"----------------------------------------------------------------------
*"*"Local interface:
*" TABLES
*" HTML STRUCTURE W3HTML OPTIONAL
*" TMAT STRUCTURE W3HTML
*"----------------------------------------------------------------------

* Debugging in RFC mode:
* call function 'SYSTEM_ATTACH_GUI'.
* break-point.

data: abaplist like abaplist occurs 1 with header line.
data: lhtml like W3HTML occurs 1 with header line.

DATA: matnr like mara-matnr.

refresh html.

* Process all material numbers, append html lists
loop at tmat.
  matnr = tmat.

* Add leading 000...
  if matnr co ' 0123456789'.
    unpack matnr to matnr.
  endif.

* Produce report
  Submit RMMMBEST
    exporting list to memory
    and return
      with MS_MATNR EQ matnr.


* get list from memory
  CALL FUNCTION 'LIST_FROM_MEMORY'
     TABLES
```

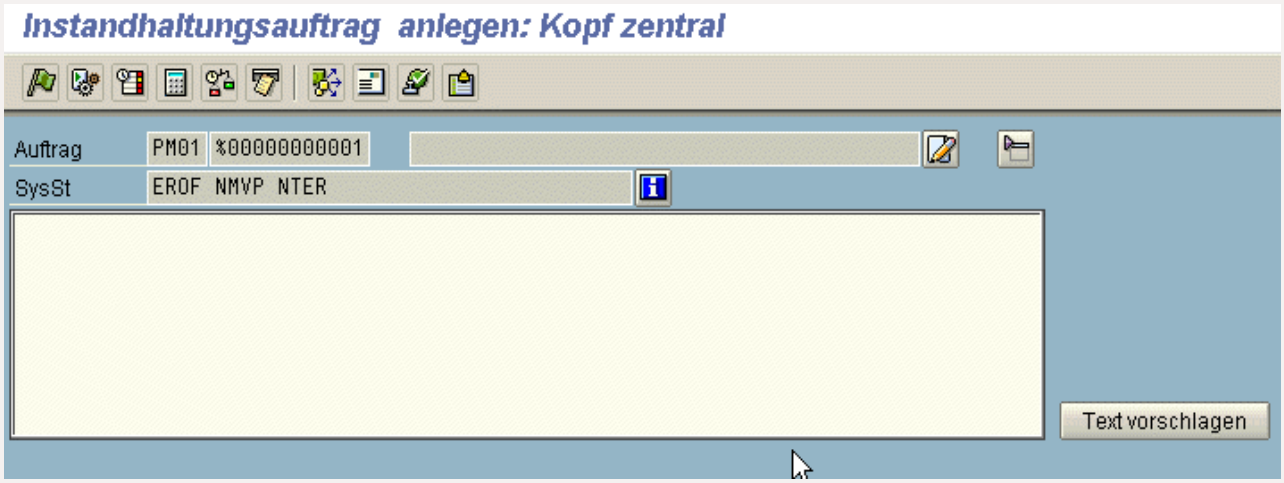
```
        LISTOBJECT = abaplist.

* and convert to html
  CALL FUNCTION 'WWW_HTML_FROM_LISTOBJECT'
     TABLES
        HTML = lhtml
        LISTOBJECT = abaplist.

  Append lines of lhtml to html.

endloop.
ENDFUNCTION.
```

11.6. Langtexte

Langtexte werden in den SAP-Transaktionen meist in einem eigenen "Text Control" erfasst, das die Eingabe beliebig vieler Zeilen mit Zeilenumbruch unterstützt. In einigen Fällen kommt der "SapScript Editor" zum Einsatz oder ist vom Benutzer aufrufbar.

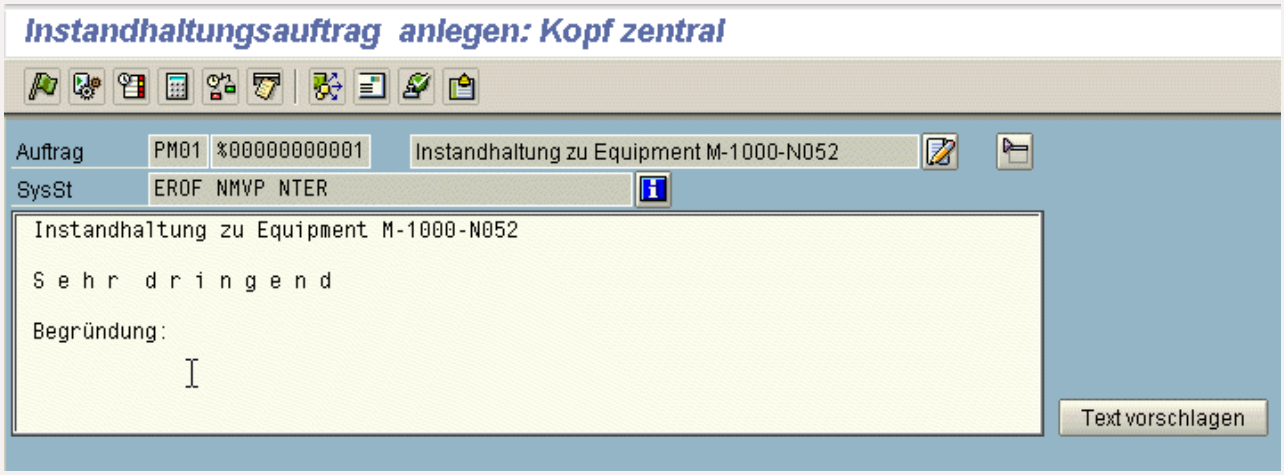
GuiXT bietet eine Reihe von Möglichkeiten zur Unterstützung der Texterfassung. Insbesondere ist es möglich, bei der Vereinfachung von Transaktionen auch die Texterfassung (ein oder mehrerer Texte) zu integrieren. Wir stellen hier die benötigten Techniken für verschiedene Anwendungsfälle dar.

1. Textvorschlag einfügen Zur Unterstützung der Texterfassung bieten Sie ein oder mehrere zusätzliche Drucktasten an, durch die ein Standardtext in das Textfenster gestellt werden kann. Dieser kann dann vom Benutzer ergänzt werden. Der Standardtext kann durch die Verwendung von Textvariablen auf die Transaktionsdaten zugeschnitten werden. Es ist auch möglich, je nach eingegebenen Daten (Priorität z.B.) einen passenden Textvorschlag einzublenden.



GuiXT Script SAPLCOIH.D3000.txt:

```
if X[LTEXT]
  Pushbutton (7,84) "Text vorschlagen" process="textvorschlag.txt"
endif
```



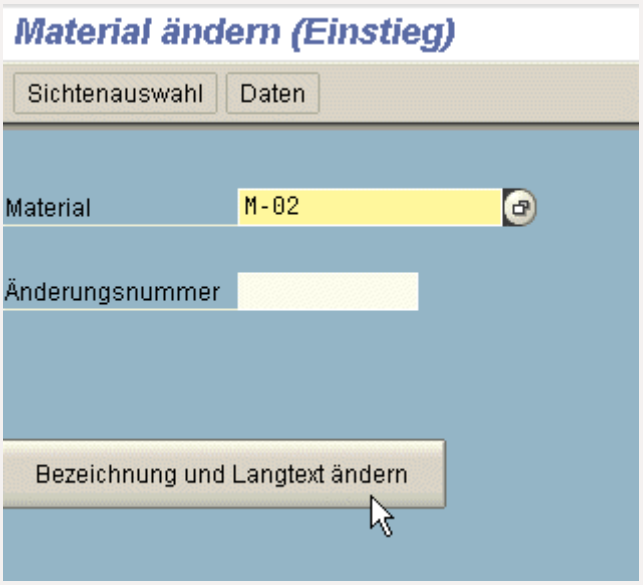
InputScript textvorschlag.txt:

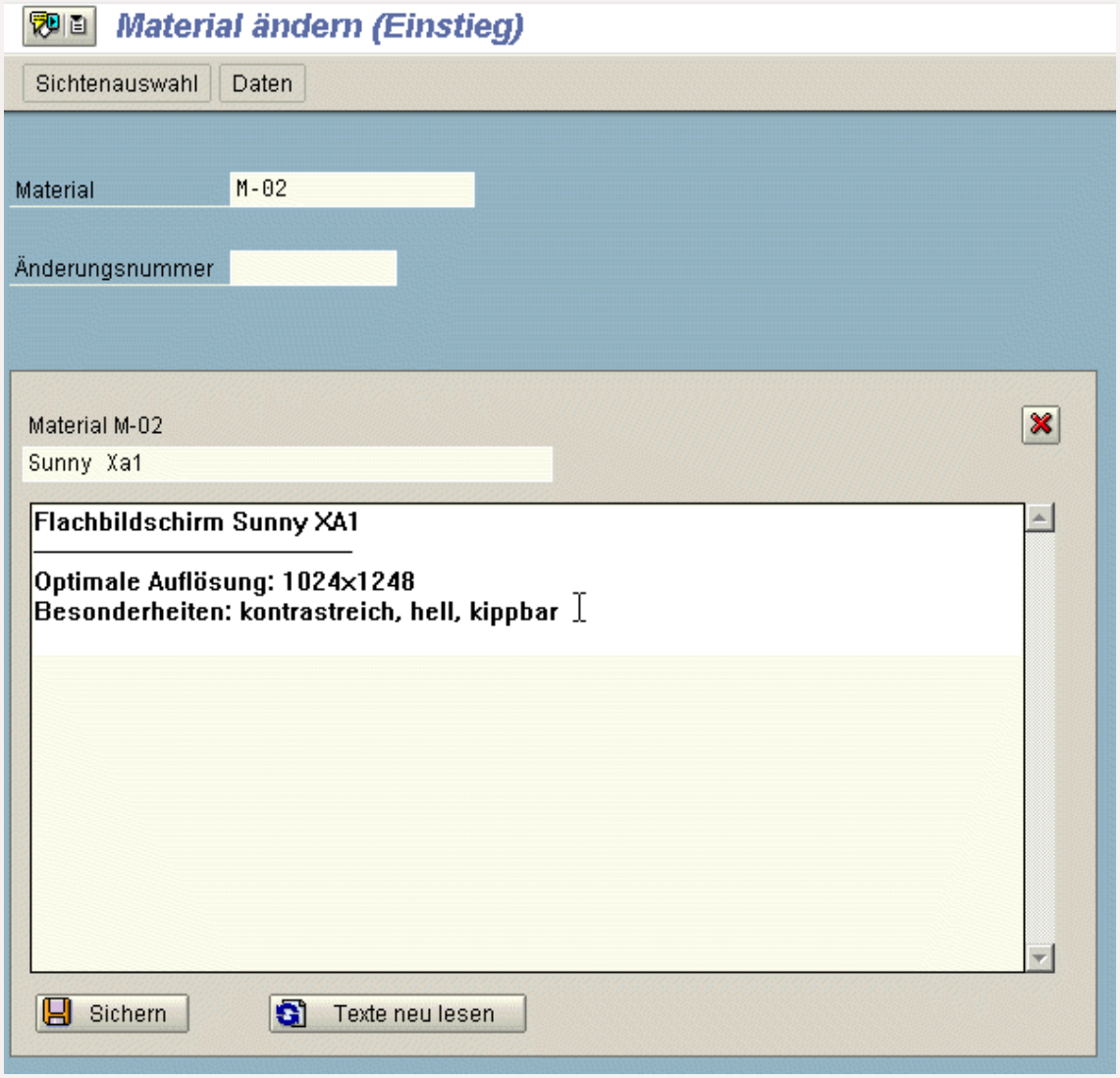
```
Screen saplcoih.3000

Set V[Prio] "&F[Priorität]"
if V[Prio=1]
  CopyText fromTemplate="iw31text_prio1.txt" toText="iw31"
else
  CopyText fromTemplate="iw31text.txt" toText="iw31"
endif
CopyText fromText="iw31" toScreen="X[LTEXT]"
Enter
```

Textvorschlag zu Priorität 1 iw31text_prio1.txt:

```
Instandhaltung zu Equipment &F[Equipment]

S e h r  d r i n g e n d

Begründung:
```

2. Texterfassung in Einbildtransaktionen Als Beispiel fügen wir in Transaktion MM02 (Ändern Materialstammsatz) eine Drucktaste ein, über die der Benutzer unmittelbar Bezeichnung und Langtext des Materials ändern kann. Das Beispiel soll die Technik demonstrieren; in konkreten Anwendungsfällen werden neben dem Langtext meist weitere Daten erfasst werden.





GuiXT Script SAPLMGMM.D0060.txt:

```
if Q[Transaction=MM02]
  Offset (6,0)
  if V[MM02_DisplayTextbox=X]
    Box (0,0) (17,82)
    InputField (1,1) "Material &V[MM02_Material]" (2,1) Size=40
name="MM02_Materialtext"
    TextBox (3,1) (15,80) name="MM02_text"

    Pushbutton (16,2)  "@2L@Sichern"           process="MM02_UpdateText.txt"
    Pushbutton (16,20) "@42@Texte neu lesen"   process="MM02_ReadText.txt"
    Pushbutton (1,78)  "@02@"                  process="MM02_CloseText.txt"

  else
    Pushbutton (1,0) "Bezeichnung und Langtext ändern" process="MM02_ReadText.txt"
size=2
  endif
endif
```

InputScript MM02_UpdateText.txt:

```
  Set F[Material] "&V[MM02_Material]"
  Enter

Screen SAPLMGMM.0070
```

```
  set cell[Table,0,1] "X"
  Enter

Screen SAPLMGMM.4004
  Enter "=PB26"

Screen SAPLMGMM.4300
  Set F[MAKT-MAKTX] "&V[MM02_Materialtext]"
  CopyText toScreen="X[LONGTEXT_GRUNDD]" fromText="MM02_text"
  Enter "/11"
```

InputScript MM02_ReadText.txt:

```
Set V[MM02_Material] "&F[Material]"

Screen SAPLMGMM.0070
  set cell[Table,0,1] "X"
  Enter

Screen SAPLMGMM.4004
  Enter "=PB26"

Screen SAPLMGMM.4300
  Set V[MM02_Materialtext] "&F[MAKT-MAKTX]"
  CopyText fromScreen="X[LONGTEXT_GRUNDD]" toText="MM02_text"

  Set V[MM02_DisplayTextbox] "X"
  Enter "/NMM02"
```

InputScript MM02_CloseText.txt:

```
Set V[MM02_DisplayTextbox] "N"
Enter "/NMM02"
```

11.7. Variablen, Berechnungen, Vergleichsoperationen

Notation

Eine Scriptvariable wird mit **V[varname]** bezeichnet, wobei varname ein frei wählbarer Name ist. Für den Namen gilt:

- Nur Buchstaben, Ziffern und den Unterstrich "_" oder Bindestrich "-" verwenden und mit einem Buchstaben oder einer Ziffer beginnen lassen
- Der Variablennamen unterscheidet Gross- und Kleinbuchstaben, d.h. V[X8] ist ungleich V[x8]

Tipp: Eine konsistente Namensgebung für die Variablen ist hilfreich. Zum Beispiel immer nur Kleinbuchstaben, bei Bedarf durch "_" getrennt, etwa V[previous_account]. Oder mit Gross- und Kleinbuchstaben:
V[PreviousAccount]. Es ist günstig, alle zu einer bestimmten Transaktion gehörenden Variablen mit dem Transaktionscode oder einer entsprechenden Kennung für mehrere Transaktionen beginnen zu lassen, z.B. V[VA01_PreviousAccount] oder [VA_PreviousAccount].

Inhalt der Variablen

Der Wert einer Variablen ist immer ein String. Der Wert wird eingesetzt, wenn in einer Scriptzeile ein Ausdruck **&V[varname]** erkannt wird. Erst nach Ersetzung aller Variablen wird die Anweisung interpretiert. Die Maximallänge der Strings intern ist 4000 Byte. Falls längere Zeichenreihen benötigt werden, bitte Langtextvariablen (siehe CopyText-Anweisung) benutzen.

Systemvariablen

Einige Variablen sind schon systemseitig vordefiniert. Ihr Name beginnt mit einem Unterstrich gefolgt von Kleinbuchstaben. Beispiele:

V[_user]
V[_client]
V[_transaction]
V[_host]
V[_database]

Sie finden eine ausführliche Liste der Systemvariablen in der Dokumentation: Systemvariablen.

Gültigkeit

Die Variablen sind global innerhalb eines SAP GUI Modus gültig. Sie sind also sichtbar und behalten ihren Wert in allen anderen Skripten, die im gleichen Modus verarbeitet werden. Wenn der Benutzer eine neue SAP GUI Session beginnt, z.B. durch direkte Eingabe des OK-Codes /OVA01, dann steht in dem neuen Modus ein neues Set von Variablen zur Verfügung.

Falls es einmal nötig sein sollte, Werte modusübergreifend zur Verfügung zu haben, können Sie mit Langtextvariablen und den Optionen toSharedText sowie fromSharedText der CopyText-Anweisung arbeiten. Häufiger kommt es vor, dass Parameter eines mit "/O..." aufgerufenen InputScripts im neuen Modus benötigt werden. Die Parameterübergabe in den neuen Modus erfolgt automatisch. Analog ist auch die Rückgabe von Werte aus einem aufgerufenen Modus möglich, siehe Anweisung "ReturnValues".

Variablen zuweisen

Durch "Set" weisen Sie einer Variable einen Wert zu. Beispiel:

Set V[MM_init] "X"

Um einer Variablen den Wert einer anderen Variablen zuzuweisen, arbeiten Sie mit der &V[...]-Notation. Beispiel:

Set V[MM_nr2] "&V[MM_nr1]"

Die Schreibweise

Set V[MM_nr2] &V[MM_nr1] // !! nicht empfehlenswert !!

ist nicht zu empfehlen, denn falls der Wert der Variablen V[MM_nr1] ein Leerzeichen enthält, wird nur der erste Teil des Strings zugewiesen und der restliche Teil als Option zum Set-Befehl interpretiert. Unproblematisch ist das Weglassen nur bei Zahlenwerten, z.B.

```
Set V[i] 1
...
Set V[i] &V[i] + 1
```

Wert einer Variablen abfragen

Mit "if" können Sie abfragen, ob eine Variable einen bestimmten Wert hat. Beispiel:

```
if V[MM_init=X]
  ....
endif
```

Vorsicht: Die Gleichheit wird hier sehr locker interpretiert :

- Gross-/Kleinschreibung wird ignoriert
- Ein Wert der nur aus Nullen "0000" besteht wird wie eine leere Zeichenreihe behandelt

Der Grund hierfür ist, dass die Variablen sehr häufig zum Vergleich eingegebener Werte benutzt werden, und hier müsste man sonst die automatische Umsetzung in Grossbuchstaben in der SAP-Anwendung bei zahlreichen Feldern sowie die Sonderbehandlung von 000..0 im Script berücksichtigen.

Sie können aber durch die Option -strict bei if erreichen, dass auf exakte Gleichheit geprüft wird:

```
if V[FI_mwskz=V0]  -strict
  ...
endif
```

Wenn Sie im Vergleich "<" oder ">" statt "=" verwenden, interpretiert GuiXT beide Ausdrücke als Zahlenwerte und vergleicht dann. In folgendem Beispiel fragen wir ab, ob die Variable k einen Wert <100 hat:

```
if V[k<100]
  ...
endif
```

Kurzform für Abfrage, ob Variable nicht leer ist

Für die häufig vorkommende Abfrage, ob eine Variable nicht leer ist, können Sie die Notation

```
if V[varname]
  ....
endif
```

verwenden.

Zwei Variablen vergleichen

Zum Vergleich zweier Variablen verwenden Sie statt eines festen Vergleichswerts den Ausdruck &V[varname]. Beispiel:

```
if V[konto1=&V[konto2]]
  ...
endif
```

Bitte beachten: zwei schliessende Klammen am Ende und keine "..." verwenden.

Generisches Rücksetzen einer Menge von Variablen

Wenn Sie bei "Set" hinter den Variablennamen einen * setzen, erhalten alle Variablen, die mit dem angegebenen Namensteil beginnen, den angegebenen Wert. Das ist praktisch zum Zurücksetzen einer Menge von Variablen, meist wenn eine neue Transaktion beginnt, was Sie über die Systemvariable V[_transactionid] abfragen können. Beispiel:

```
if not V[_transactionid=&V[MM_transactionid]]

  Set V[MM_*] ""
  Set V[MM_transactionid] "&V[_transactionid]"

endif
```

Variablen konkatenieren

Bei "Set" und jeder anderen Anweisung können Sie mehrere Variablenwerte durch Hintereinanderschreiben zusammenfügen. Beispiel:

```
Set V[MM01_text] "&V[M01_text1]&V[MM01_text2]"
```

Dazwischen können sich auch konstante Strings befinden. Beispiel:

```
Set V[id] "&V[a]-&V[b]-&V[c]"
```

Teilstrings abgreifen

Sie können aus einer Variablen Teilstrings entnehmen. Die Notation ist dabei jeweils, dass Sie in runden Klammern dahinter die erste und die letzte gewünschte Position angeben, durch Bindestrich oder Komma getrennt. Beispiel:

```
Set V[x] "00113100"
Set V[y] "&V[x](5-6)"
```

Jetzt hat V[y] den Wert "31". Ebenso bei

```
Set V[x] "00113100"
Set V[y] "&V[x](5,6)"
```

Es gibt zwischen den beiden Notationen einen Bedeutungsunterschied, der sich nur in Unicodesystemen bemerkbar macht. GuiXT arbeitet intern dann mit der UTF8-Codierung, und dort belegen manche Zeichen, zum Beispiel Umlaute ä,ö,ü, mehr als 1 Byte. Während (4-6) die Bytes 4 bis 6 abgreift, bezieht sich (4,6) auf die Zeichen 4 bis 6. Beispiel:

```
Set V[x] "München"
Set V[y] "&V[x](4-6)"
```

Jetzt hat V[y] den Wert "nch". Dagegen bei

```
Set V[x] "München"
Set V[y] "&V[x](4,6)"
```

den Wert "che". Da in Variablen auch Strukturen abgelegt werden können, deren Komponenten aus binären oder gepackten Zahlen bestehen, kann nicht generell die Zeichen-Interpretation verwendet werden.

Länge eines Strings in Bytes oder in Zeichen

Die Option -stringlength bei Set liefert die Länge einer Zeichenreihe in Bytes, während -charcount die Anzahl der darin enthaltenen Zeichen liefert. In Unicode-Systemen können die beiden Werte unterschiedlich sein.

Beispiel:

```
Set V[leng] "München" -stringlength

Set V[count] "München" -charcount
```

V[leng] ist 7 in nicht-Unicodesystemen und 8 in Unicodesystemen, da 'ü' in UTF8 zwei Bytes belegt. Dagegen ist V[count] immer 7.

Tipp: Verwenden Sie -stringlength in Verbindung mit der Teilstringnotation (i-k) und -charcount in Verbindung mit der Teilstringnotation (i,k); siehe Beispiel im nächsten Abschnitt.

Teilstrings mit variablen Positionen

Die Positionsangaben für Teilstrings können variable sein, also zum Beispiel (&V[i],&V[k]) statt (5,8), wobei V[i] und V[k] die gewünschten Positionen als Ziffern enthalten. In dem folgenden Beispiel löschen wir in einem gegebenen String alle Punkte; wir verwenden Rechenoperationen für Variablen (siehe unten). Der String V[in] wird Zeichen für Zeichen auf '.' untersucht und dabei der Ausgabestring V[out] aufgebaut.

```
// zum Testen
Set V[in] "12.580.720.4"

Set V[out] ""
Set V[charcount] "&V[in]" -charcount
Set V[k] 1

label next_char

if V[k>&V[charcount]]
```

```
  goto done
endif

Set V[char] "&V[in](&V[k],&V[k])"

if not V[char=.]
  Set V[out] "&V[out]&V[char]"
endif

Set V[k] &V[k] + 1

goto next_char
```

Komponenten aus SAP-Strukturen abgreifen

Beim Aufruf von Funktionsbausteinen erhalten Sie häufig Informationen als SAP-Struktur, aus der Sie die einzelnen Komponenten entnehmen müssen. Hier ist das Arbeiten mit festen Offsets und Längen sehr fehleranfällig. GuiXT unterstützt die symbolische Notation über eine SAP Dictionary-Struktur. Sie nennen hinter der Variablen in runden Klammen die SAP Dictionary-Struktur und dann, durch Bindestrich abgetrennt, den Namen der Komponente. Als Beispiel lesen wir die Adressdaten zu einer Kundennummer und greifen den Ort ab:

```
Call "BAPI_CUSTOMER_GETDETAIL2" in.CUSTOMERNO="0000002000" out.CUSTOMERADDRESS="addr"
Set V[city] "&V[addr](BAPICUSTOMER_04-CITY)"
```

Erläuterung: Der Funktionsbaustein "BAPI_CUSTOMER_GETDETAIL2" besitzt einen Export-Parameter "CUSTOMERADDRESS". Dieser wird in unsere Variable V[addr] gestellt. Laut SAP Funktionsbibliothek ist die Dictionary-Struktur des Parameters "CUSTOMERADDRES" "BAPICUSTOMER_04", und dort befindet sich der Ort in der Komponente "CITY".

Häufig liefern Funktionsbausteine auch Tabellen zurück, bei denen jede Zeile eine SAP-Struktur besitzt, auf die wir mit der symbolischen Notation zugreifen können.

Komponenten in SAP Strukturen setzen

Analog können Sie in einer Variable Teilstrings setzen. Dabei können Sie mit festen Offsets und Längen arbeiten oder mit symbolischen Namen laut SAP-Dictionary. Als Beispiel lesen wir alle Kunden mit Nummern zwischen 1 und 50 über den Funktionsbaustein BAPI_CUSTOMER_GETLIST, der als Eingabe eine Tabelle erwartet, in der jede Zeile ein Kundennummernintervall enthält.

```
Set V[line](BAPICUSTOMER_IDRANGE-SIGN) "I"
Set V[line](BAPICUSTOMER_IDRANGE-OPTION) "BT"
Set V[line](BAPICUSTOMER_IDRANGE-LOW) "0000000001"
Set V[line](BAPICUSTOMER_IDRANGE-HIGH) "0000000050"

CopyText fromString="line" toText="idrange"

Call "BAPI_CUSTOMER_GETLIST" table.IDRANGE="idrange" table.ADDRESSDATA="adrtab"
```

Berechnungen

Die **Set** Anweisung unterstützt die Grundoperatoren + , -, *, /. Alle Operanden werden zunächst in Zahlen umgewandelt. Das Ergebnis wird dann wieder in einem Stringformat gespeichert. Das Ergebnis wird, falls nicht ganzzahlig, auf 2 Dezimalstellen kaufmännisch gerundet ausgegeben; dabei können Sie eine andere Anzahl von Dezimalstellen setzen (decimals=) sowie weitere Optionen, siehe Set-Anweisung.

```
Set V[meineVariable] Wert1 + Wert2
Set V[meineVariable] Wert1 - Wert2
Set V[meineVariable] Wert1 * Wert2
Set V[meineVariable] Wert1 / Wert2
```

Die Rechnungen erfolgen intern über doppeltgenaue Gleitpunktzahlen.

Beispiel 1

```
Set V[x]  2  /  3 decimals=5
```

Resultat: 0,66667

Datumsangaben

Bei der Umwandlung in Zahlen werden Datumsangaben wie z.B. "01.06.2014" erkannt und automatisch in Tagen berechnet. Das Ergebnis wird wieder als Datum in gleichem Format ausgegeben, falls von einem Datum Zahlen addiert oder subtrahiert wurden.

Beispiel 1:

```
Set V[x]  "01.06.2014" - 1
```

Resultat: "31.05.2014"

Beispiel 2:

```
Set V[x]  "01.06.2014" -  "01.01.2014"
```

Resultat: "151"

Beispiel 3:

Die folgende Datumsrechnung prüft, ob ein gegebenes Datum V[date] ein Sonntag ist. Dazu wird ein beliebiger Sonntag, z.B. der 31.12.2000, von dem Datum subtrahiert und dann berechnet, ob die Differenz durch 7 teilbar ist:

```
Set V[x] "&V[date]" - "31.12.2000"
Set V[y] &V[x] / 7 decimals=0
Set V[y] &V[y] * 7

if V[x=&V[y]]
  Message "Sonntag"
endif
```

Variablen im Namen einer Variablen

Sie können in einem Variablennamen den Inhalt anderer Variablen verwenden. Beispiel:

```
Set V[name] "x"
Set V[&V[name]] "10"
```

Hierdurch wird V[x] auf "10" gesetzt.

Meist benutzt man diese Möglichkeit für einen fortlaufenden Index. Als Beispiel übertragen wir aus den angezeigten Zeilen einer Erfassungstabelle "Alle Positionen" die Spalte "Material" in fortlaufend nummerierte Variablen:

```
Set V[i] 1

label start

Set V[VA_material_&V[i]]  "&cell[Alle Positionen,Material,&V[i]]"
if not V[VA_material_&V[i]]
  goto end
endif


Set V[i] &V[i] + 1
goto start

label end
```

Umgebungsvariablen

Den Wert einer Windows Umgebungsvariablen erhalten Sie über die Notation "&%[xxx]" wobei xxx der Name der Umgebungsvariable ist.

Wenn Sie zum Beispiel eine temporäre Datei generieren wollen, kann es sinnvoll sein, den Ordner aus der Umgebungsvariable TEMP zu entnehmen:

```
Set V[myFilename] "&%[TEMP]\guixtva01.txt"
```

© by Synactive GmbH

11.8. Externe Datenquellen integrieren

Es gibt verschiedene Möglichkeiten, mit GuiXT externe Datenquellen in das R/3 System zu integrieren. Wir beschreiben hier die unterschiedlichen Methoden zusammen mit ihren Vor- und Nachteilen:

A Sie zeigen ein kleines Bild auf dem R/3 Bildschirm an. Wenn der Benutzer auf das Bild klickt, wird ein Programm gestartet, das die zusätzlichen Daten in einem separaten Fenster anzeigt. Die Programmparameter werden aus der aktuellen R/3 Maske übernommen.

B Sie gebrauchen den GuiXT html Viewer und rufen ein Programm (z.B. CGI) auf einem Server im Intranet oder Internet auf, das die externen Daten liest und eine html Seite mit diesen Daten zurück sendet.

C Sie gebrauchen die GuiXT `call` Anweisung, um eine dll Funktion aufzurufen. Diese dll Funktion liefert einige Strings zurück, die Sie mit der GuiXT `Text` Anweisung direkt im R/3 Bildschirm anzeigen können. Die dll Funktion könnte auch eine html oder eine rtf Datei generieren, die Sie dann mit dem GuiXT Viewer in den Bildschirm einbinden können.

D Sie zeigen Ihren eigenen Pushbutton auf dem R/3 Bildschirm an. Wenn der Benutzer den Pushbutton drückt, führen Sie ein InputScript aus (siehe InputAssistant). Das InputScript kann verschiedene Aufgaben ausführen: Daten aus anderen R/3 Masken übernehmen, ABAP über RFC aufrufen, dll Funktionen aufrufen. Das InputScript liefert einige GuiXT Variablen, die mit der Text Anweisung oder mit der `Template` Funktion des GuiXT Viewers angezeigt werden können.

Sie brauchen die folgenden GuiXT Komponenten:

A GuiXT

B GuiXT + Viewer

C GuiXT (evtl. auch den Viewer)

D GuiXT + InputAssistant (evtl. auch den Viewer)

A Mit Hilfe eines Bildes

Beispiel

```
Image (20,1) "info.gif" Start=\\P450\prog1\sqlread1.exe&[Material]"
```

Vorteile

- Sie können eine Programmiersprache Ihrer Wahl für das .exe Programm benutzen.
- Die Anzeige der externen Daten ist völlig unabhängig vom R/3 Bildschirmlayout.
- Das neue Fenster bleibt sichtbar, auch wenn der Benutzer den aktuellen R/3 Bildschirm verläßt.

Nachteile

- Der Benutzer muß zuerst auf das Bild klicken.
- Sie können keine Werte verwenden, die der Benutzer gerade eben eingegeben hat, ohne zuvor "Enter" zu drücken. (Sie werden immer den letzten Wert bekommen, den R/3 in diesem Feld angezeigt hat.)

B Mit Hilfe des GuiXT html Viewers

Beispiel

```
View (16.1)(24,120)"template=\\p450\html\mat1.html"
```

Alle &[...] Variablen werden in der html Datei durch ihre aktuellen Werte ersetzt. Sie können auch URLs in der html Datei verwenden, die diese Werte an CGI Programme übergeben und Sie können in der html Datei auch Java Script verwenden, das mit R/3 Werten arbeitet.

Vorteile

- Die html Anzeige wird direkt in den R/3 Bildschirm integriert.
- Sie können html Werkzeuge für das Layout verwenden und Sie können Java Script in der html Datei verwenden für anspruchsvolle Aufgaben.
- Sie können dem Benutzer weiterführende Aktionen anbieten (innerhalb html oder auch zurück zu R/3) und auch Hyperlinks zum Internet.
- Der Benutzer kann lokal ausdrucken, was er sieht.

Nachteile

- Sie müssen die notwendigen html Techniken kennen, um externe Daten aus html einzulesen.

Ein Beispiel finden Sie im Viewer Tutorial.

C Mit Hilfe eines dll Aufrufs

Beispiel

```
Call "extinfo" "dll=gxexit" "In=&[Material]" Out=Info1 Out=Info2 Out=Info3
Text (16,1) "External Info1"
Text (16,30) "&[Info1]"
Text (17,1) "External Info2"
Text (17,30) "&[Info2]"
Text (18,1) "External Info3"
Text (18,30) "&[Info3]"
```

oder

```
Call "extinfo" "dll=gxexit" "In=&[Material]" Out=Filename
View (16,1) (24,120) "&[Filename]"
```

Vorteile

- Die Anzeige ist direkt in den R/3 Bildschirm integriert.

- Sie können html oder rtf für die Anzeige verwenden.
- Sie können dem Benutzer weiterführende Aktionen anbieten (innerhalb html oder auch zurück zu R/3) und auch Hyperlinks zum Internet.
- Der Benutzer kann lokal ausdrucken, was er sieht.

Nachteile

- Sie müssen sich mit dll auskennen (normalerweise mit C++, VB ist auch möglich).

Siehe auch den Special Topic Artikel **dll Ausgabefunktionen**

D Mit Hilfe eines InputScripts

Beispiel

```
Pushbutton (16,1) "Externe Daten" "Process=MatExtData.txt"
   using Material = [Material]
if V[MatExtData=X]
   View (16,20) (24,120) "template=MatExtData.html"
endif
```

In diesem Fall würde das InputScript einige Variablen setzen, die in der Template Datei verwendet werden. Das InputScript kann auch verschiedene Techniken wie dll Aufruf verwenden, um die dll Variablen zu setzen.

Vorteile

- Die Anzeige ist direkt in den R/3 Bildschirm integriert.
- Sie können html oder rtf für die Anzeige verwenden.
- Sie können dem Benutzer weiterführende Aktionen anbieten (innerhalb html oder auch zurück zu R/3) und auch Hyperlinks zum Internet.
- Der Benutzer kann lokal ausdrucken, was er sieht.
- Sie können zusätzliche R/3 Informationen und externe Datenquellen mischen.
- Sie bekommen die aktuellen Werte, die der Benutzer gerade eingegeben hat.

Nachteile

- Sie müssen sich mit dll Aufrufen und evtl. auch mit ABAP Funktionen auskennen, wenn Sie sie im InputScript verwenden wollen.
- Der Pushbutton ruft immer eine Funktion im R/3 auf (default: enter)

11.9. Die Behandlung von Dateien in InputScripts

Es gibt eine Reihe von sehr nützlichen Dingen, die Sie mit Dateien in InputScripts tun können.

- Sie können Daten aus einer Datei lesen und sie in R/3 eingeben.
- Sie können Daten aus R/3 über die Transaktionen extrahieren und die so generierten Dateien in Excel oder ähnlichen Programmen verwenden.
- Sie können eine html Datei generieren und diese in R/3 anzeigen, indem Sie den GuiXT Viewer verwenden.

Wenn Sie solche Anwendungen planen, dann beachten Sie bitte, dass diese Methode sehr gut für 10 bis 100 Datensätze in einer Datei funktioniert, dass aber andere Methoden (Batch input, ABAP Programme) möglicherweise eine bessere Wahl bei größeren Datenmengen sind. Die "Scriptingmethode", die hier beschrieben ist, ist geeignet, um 50 Materialsätze über eine Exceltabelle zu verändern oder um Daten aus 120 Kostenstellen zu extrahieren. Besonders für adhoc Aufgaben stellt der InputAssistant eine leicht zu handhabende Möglichkeit dar, um eine Menge Zeit zu sparen, die Sie andernfalls für die Ausführung per Hand oder für die Programmierung verwenden müßten.

Übersicht

Wir verwenden die folgenden Anweisungen des InputAssistant:

- öffnen und schließen einer Datei: **`OpenFile, CloseFile`**;
- Werte aus einer Datei in Variablen einlesen: **`ReadFile`**;
- einen neuen Satz in eine Datei schreiben: **`AppendFile`**;
- mit Variablen arbeiten: **`Set, if`**;
- Ablaufkontrolle innerhalb eines InputScripts: **`Screen, Enter, goto, label, return`**.

Beispiel 1

- Wir nehmen an, dass wir eine Textdatei mit Materialnummern haben. Wir möchten eine zweite Datei erstellen, die die Materialnummer zusammen mit der Sparte, Warengruppe und Berechtigungsgruppe für jedes Material enthält.

  Zuerst brauchen wir einen Ort, wo wir das Script beginnen können, z.B. das R/3 Basismenü

```
MENUS00.E0040.txt:
Pushbutton (toolbar) "Materialdatei erstellen" "/NMM03"
"Process=Materialdatei_erstellen.txt"
```

  Das InputScript `Materialdatei_erstellen.txt` sieht folgendermaßen aus::

```
// Materialdatei erstellen (Beispiel für R/3 Rel. 4.5)

Parameter mat_file1 "C:\GuiXT\Input.txt"
Parameter mat_file2 "C:\GuiXT\Output.txt"


// Start
```

```
Screen SAPLMGMM.0060
OpenFile "&[mat_file1]"
OpenFile "&[mat_file2]" "-Output"

label Read_File
ReadFile "&[mat_file1]" mat_matnr

if not V[mat_matnr]
CloseFile "&[mat_file1]"
CloseFile "&[mat_file2]"
Return "Materialdatei wurde generiert" "&[mat_file2]"
endif

Set [Material] "&[mat_matnr]"
Enter

// Bild auswählen
Screen SAPLMGMM.0070
Set C[Grunddaten 1] "X"
Enter

// Grunddaten 1
Screen SAPLMGMM.4000
Set V[mat_division] "&[Sparte]"
Set V[mat_matgroup] "&[Warengruppe]"
Set V[mat_autgroup] "&[Berechtigungsgruppe]"

AppendFile "&[mat_file2]" mat_matnr mat_division mat_matgroup mat_autgroup

Enter "/NMM03"

// MM03 Neustart, zum Beginn des Skripts gehen
Screen SAPLMGMM.0060
goto Read_File
```

Die Laufzeit für 100 Materialnummern betrug 67 Sekunden in unserem Testsystem (zentrales System auf einem PC mit 266 MHz Pentium und 256 MB RAM.)

Example 2

Angenommen wir haben eine Textdatei mit Daten, die GL Dokumente darstellen. Wir möchten dieses Dokumente mit Transaktion FB01 nach R/3 übernehmen. Wenn es Fehlermeldungen der R/3 Transaktion gibt, dann werden diese in ein Logfile gespeichert, zusammen mit weiteren Daten.

MENUS00.E0040.txt:

```
Pushbutton (toolbar) "GL upload" "/NFB01" "Process=FB01_GL.txt"
```

Das InputScript **FB01_GL.txt** lautet folgendermaßen:

```
// Dateinamen
Parameter FB01FILE "C:\GL Test\FB01_data.txt"
Parameter FB01ERR  "C:\GL Test\FB01_err.txt"

// Start der Dateiausführung
OpenFile "&[FB01FILE]" "Delimiter=;"
OpenFile "&[FB01ERR]" "-Output"

Set V[FB01_errors] 0
Set V[FB01_records] 0
```

```
Screen SAPMF05A.0100

label start

// Gibt es eine Fehlermeldung im vorherigen Datensatz?
if V[_lasterror]

  // Fehlermeldungszähler hochsetzen
  Set V[FB01_errors] &[FB01_errors] + 1

  // Fehler Logfile schreiben
  AppendFile "&[FB01ERR]" s_date s_text s_pstky1 s_account1 s_amount1 s_pstky2
s_account2 s_amount2 _lasterror

  // Fehlerzähler zurücksetzen (!). Automatisches Rücksetzen nur bei Start des
InputScripts
  Set V[_lasterror]
endif

ReadFile "-StripQuotationMarks" "&[FB01FILE]" s_date s_text s_pstky1 s_account1
s_amount1 s_pstky2 s_account2 s_amount2

// Ist ein weiterer Datensatz auszuführen?
if V[s_date]

  // Zähler hochsetzen
  Set V[FB01_records] &[FB01_records] + 1

  Set [Document date] &[s_date]
  Set [Doc.header text] &[s_text]
  Set [Company Code] "0001"
  Set [Type] "SA"
  Set [Currency/rate] "USD"

  Set [Pstky] &[s_bschl1]
  Set [Account] &[s_account]

  Enter "OnError=/NFB01"

else

  // Ausführung beendet, Dateien schließen
  CloseFile "&[FB01FILE]"
  CloseFile "&[FB01ERR]"

  if V[FB01_errors=0]
    Return "Processing done, no errors, &[FB01_records] documents in total"
  else

    // Fehlermeldungen in Error log anzeigen
    View "&[FB01ERR]"

    Return "Processing done: &[FB01_errors] errors, &[FB01_records] documents in total"
  endif
endif

Screen SAPMF05A.0300
Set [Amount] &[s_amount1]
Set [PstKy]  &[s_pstky2]
Set [GL/ account]  &[s_account2]

Enter "OnError=/NFB01"


Screen SAPMF05A.0300
```

```
Set [Amount] &[s_amount2]
Enter "/11" "OnError=/NFB01"

Screen SAPMF05A.0100
goto start
```

Die Datei **FB01_data.txt** könnte folgendermaßen aussehen:

```
01122000;"GL text1";40;113100;100;50;113101;100
02122000;"GL text2";40;113100;200;50;113777;200
15122000;"GL text3";40;113100;300;50;113101;300
32122000;"GL text4";40;113100;400;50;113101;400
14122000;"GL text5";40;113100;500;50;113101;500
01122000;"GL text6";40;113100;100;50;13101M;100
02122000;"GL text7";40;113100;200;50;113777;200
15122000;"GL text8";40;113100;300;50;113101;300
32122000;"GL text9";40;113100;400;50;113101;400
```

11.10. Löschen von Menueinträgen

Mit der `del` Anweisung können Sie Menüeinträge und Untermenüs löschen. Das ist nicht schwierig. Aber da es Auswirkungen auf die Symbolleiste, die Funktionstasten und die Untermenüs hat, erklären wir den Vorgang hier im Detail.

Betrachten Sie zum Beispiel Transaktion MB1A in R/3 Rel. 4.6C. Hier ist ein Teil des Hauptmenüs:

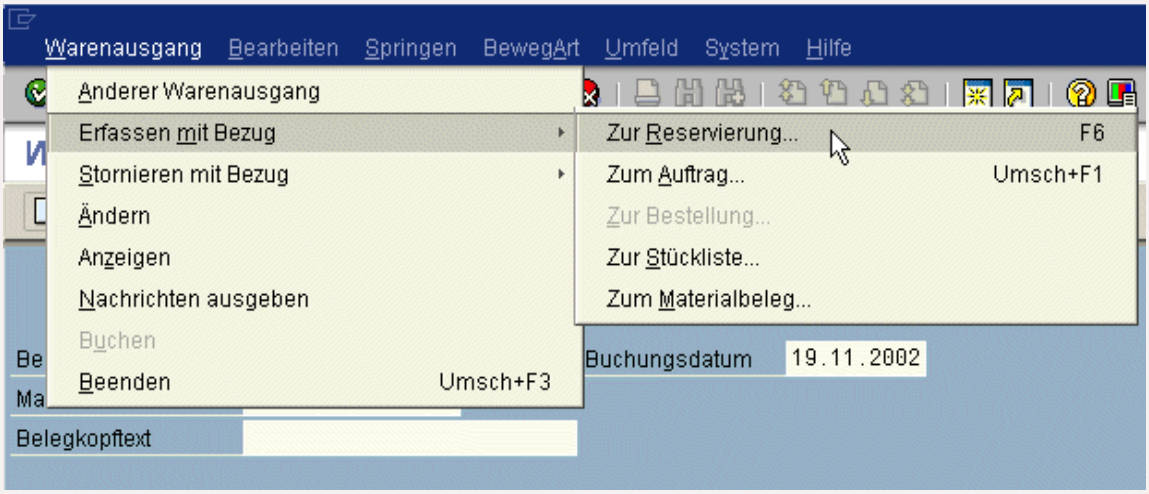


Fig. 1: Menü in der Transaktion MB1A

Um den Menüeintrag "Zur Reservierung..." zu löschen, reicht es zu schreiben

```
del M[Zur Reservierung...]
```

Das Menü sieht dann folgendermaßen aus:

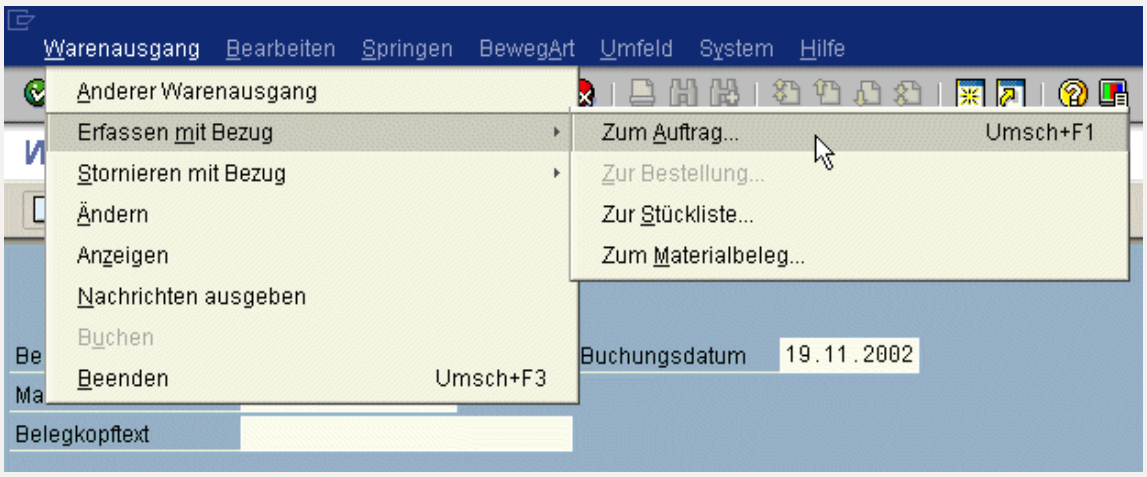


Fig. 2: Menüeintrag [Zur Reservierung...] gelöscht

Wenn es in der Symbolleiste einen Button gibt, der die gleiche Aktion ausführt wie der Menüeintrag, dann löscht GuiXT auch diesen, zusammen mit der zugeordneten Funktionstaste:

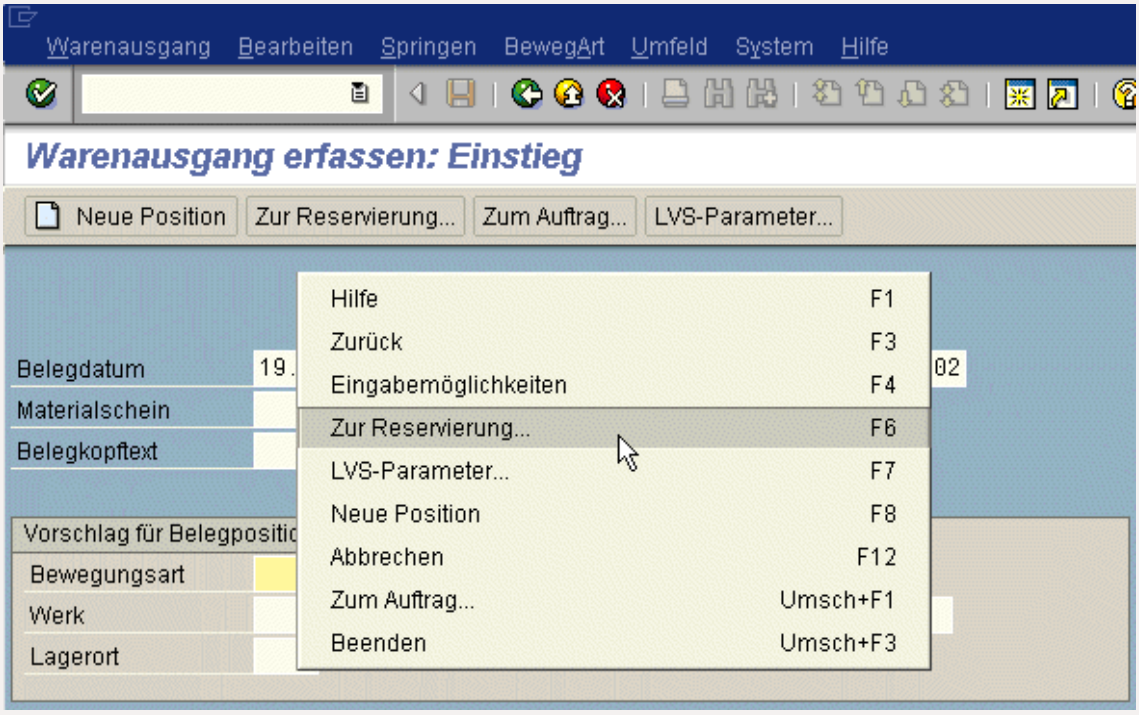


Fig. 3: Symbolleiste und Funktionstasten in MB1A, kein GuiXT Skript aktiv

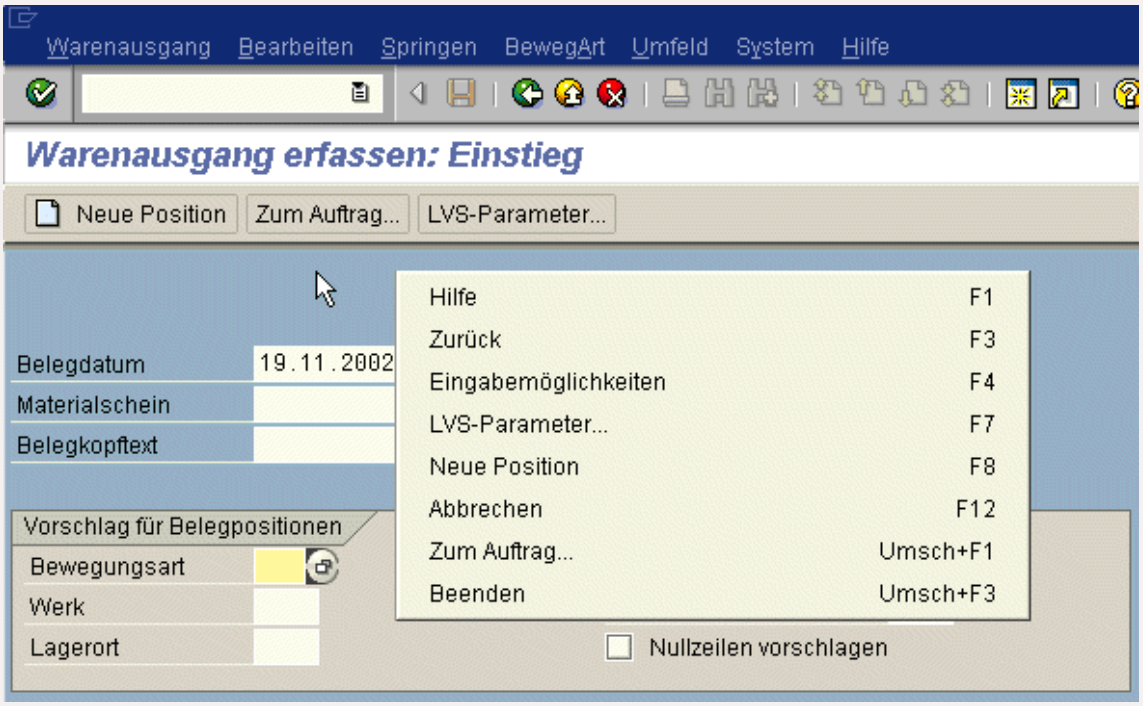


Fig. 4: Das Skript del M[Zur Reservierung...] löscht automatisch den Button und die zugehörige Funktionstaste zusammen mit dem Menüeintrag

Es kommt vor, dass mehrere Menüeintrage mit dem gleichen Text existieren. In der Transaktion MB1A ist dies der Fall für den Eintrag [Zur Reservierung...], der sowohl in dem Untermenü "Erfassen mit Bezug" als auch in "Stornieren mit Bezug" auftaucht:

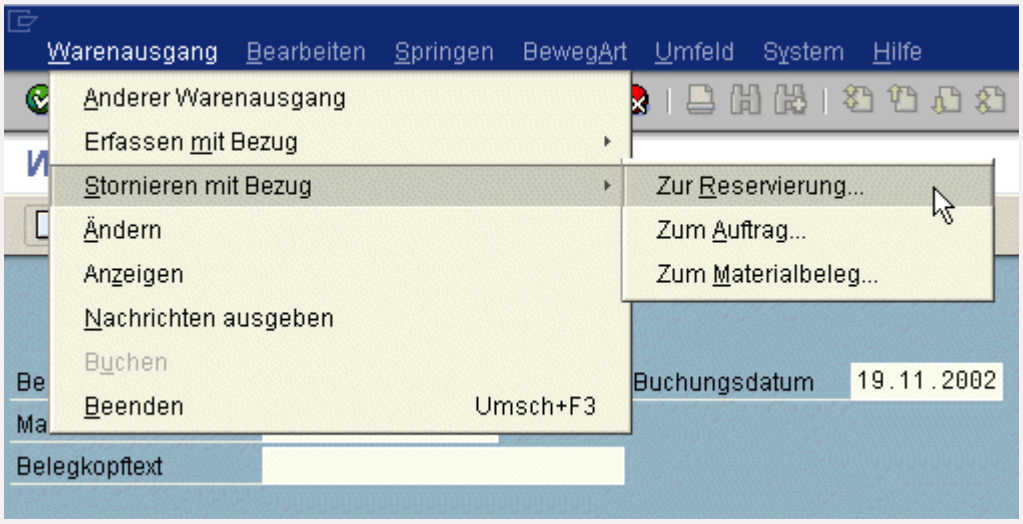


Fig. 5: Der Eintrag [Zur Reservierung...] existiert in einem zweiten Untermenü

GuiXT hat nur den ersten Eintrag im "Erfassen mit Bezug" Untermenü gelöscht. Um zwischen den beiden gleichlautenden Einträgen zu unterscheiden, verwenden Sie am besten den Funktioncode. Sie finden ihn, indem Sie F1 nach der Markierung des Menüeintrags drücken.

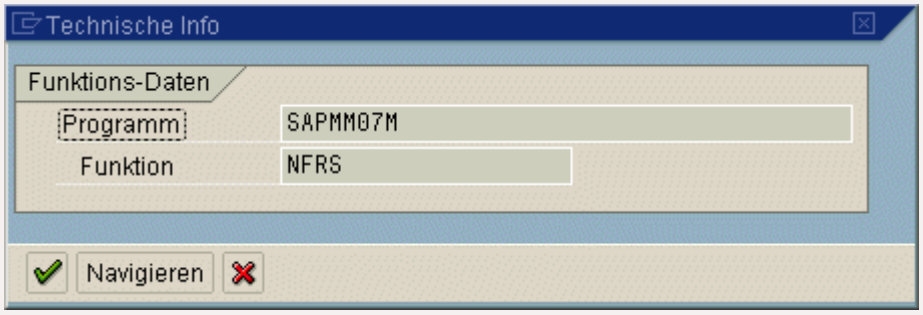


Fig. 6: Verwenden Sie F1 um den Funktionscode für einen Menüeintrag zu finden

Verwenden Sie dann die `del` Anweisung mit dem Funktionscode anstelle des Menüeintrags:

```
del M[NFRS]
```

Anwender können übrigens jeden Funktionscode direkt im OK Eingabefeld eingeben, anstelle den Menüeintrag, den Button oder die Funktionstaste auszuwählen. Wenn Sie also einen Menüeintrag durch ihr GuiXT Skript löschen, so gäbe es noch diesen Geheimweg, um die verborgene Funktion auszuwählen. Aus diesem Grund löscht GuiXT alle Funktionscodes im OK Eingabefeld, wenn sie dem gelöschten Menüeintrag entsprechen.

Hierarchische Untermenüs können ebenfalls gelöscht werden, indem Sie den Menünamen verwenden. In diesem Fall werden alle Einträge im Untermenü (und im Untermenü des Untermenüs....) gelöscht, wieder zusammen mit den Buttons in der Symbolleiste und den zugeordneten Funktionstasten.

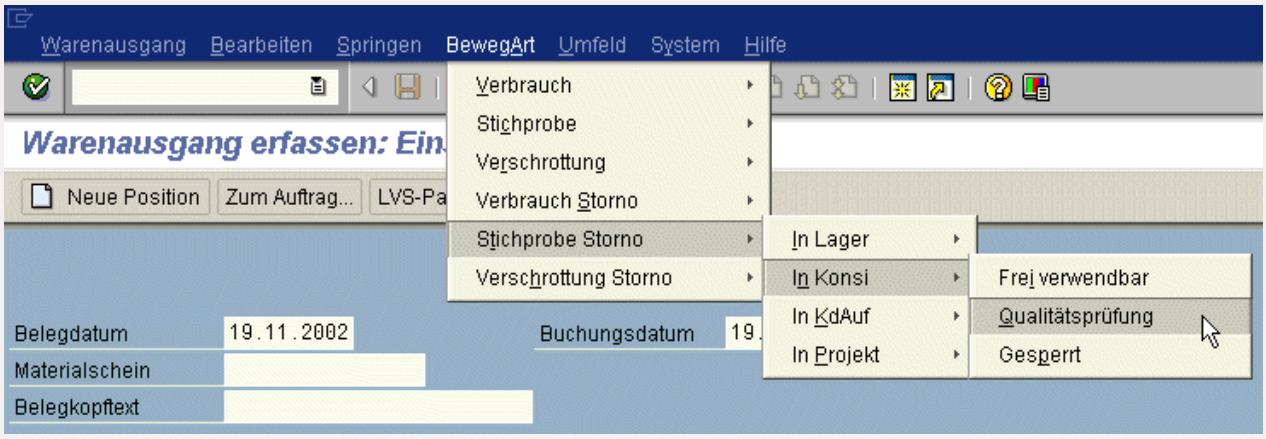


Fig. 7: Hierarchisches Untermenü

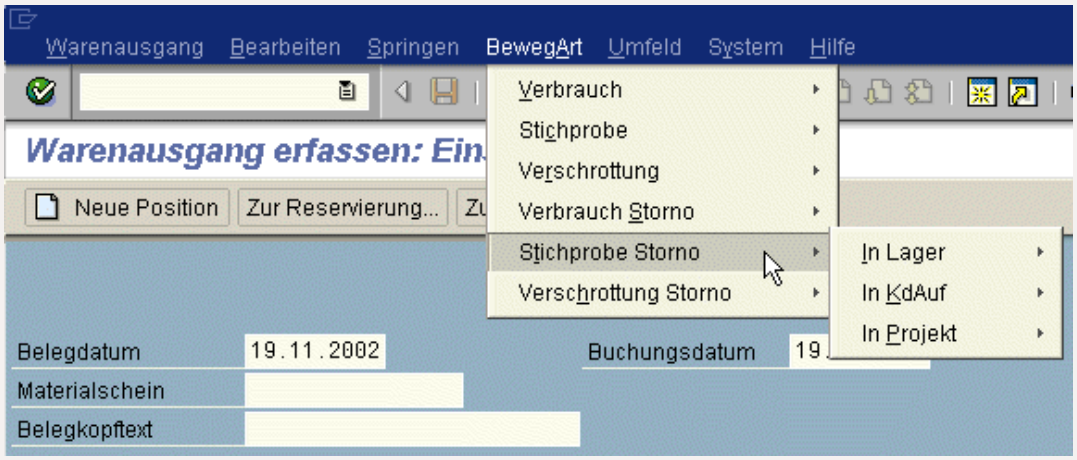


Fig. 8: Hierarchisches Untermenü nach dem Skript `del M[In Konsi]`

Hauptmenüs (wie "Warenausgang", "Bearbeiten") können nicht gelöscht werden. Selbst wenn Sie alle Menüeinträge löschen, bleibt der Menütitel erhalten, das Menü ist dann aber leer. In dem "System" und "Hilfe" Menü können Sie weder Einträge noch Untermenüs löschen.

In seltenen Fällen haben sogar Untermenüs den gleichen Namen. Das ist beispielsweise in MB1A der Fall mit dem Untermenü [In Konsi], das sowohl in [Stichprobe Storno] als auch in [Verschrottung Storno] existiert. In diesem Fall können Sie nicht wie oben beschrieben den Funktionscode verwenden, denn Untermenüs haben keinen Funktionscode. In solchen sehr seltenen Fällen können Sie mit den Positionen der Menüeinträge arbeiten, M[i,j,k,l],. Hier sind i,j,k,l bis zu vier Zahlen, die die Position in der Menühierarchie bezeichnen, beginnend mit 1. In unserem Beispiel wäre dies

```
del M[4,6,2]
```

Erklärung: 4=BewegArt, 6=Verschrottung Storno, 2=In Konsi

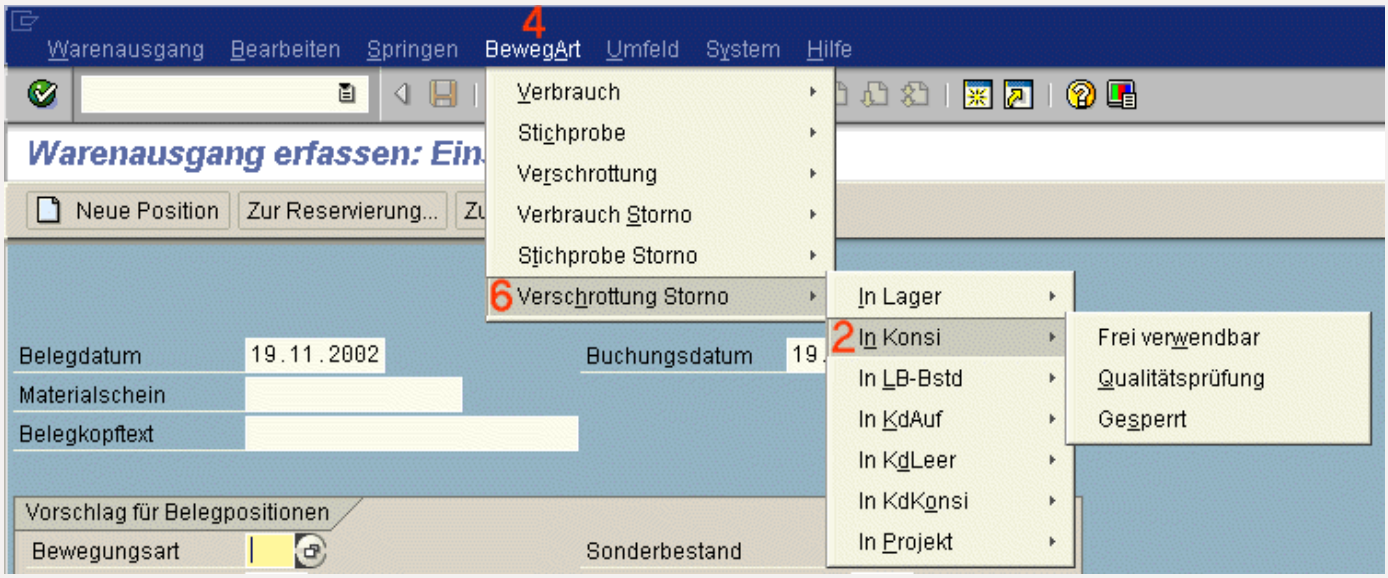


Fig. 9: Position des Untermenüs in der Menühierarchie

Ein abschließender Tipp: Falls Sie tatsächlich mit solchen absoluten Positionen arbeiten müssen, dann zählen Sie die Postionen, bevor irgendeine `del M[...]` Anweisung aktiv ist.

11.11. SAPGUI Scripting und GuiXT

With SAP GUI 6.20, SAP introduced a "scripting API": an object model that represents a screen with its controls at runtime. In WIndows environment, the API can be used from any tool that supports the COM object interface, especially from the VBScript and JavaScript languages. The object model allows to automate all user actions. In distinction to GuiXT, the SAP GUI scripting approach is not intended to change the screen layout. It completes the GuiXT functionality, allowing to manipulate special controls (like the grid control) that GuiXT cannot handle directly, but is restricted to actions that a user is able to do as well.WIth GuiXT 2004 Q2 3 and above, you can combine the two approaches. Use the new keyword "ApplyGuiScript" in a GuiXT script or an InputScript in order to perform further actions on the screen via a VBScript.We now discuss the general procedure, and we will demonstrate it with a few examples.

11.11.1. Anforderungen

- SAP GUI 6.20 and upwards
- GuiXT 2004 Q2 2 and upwards
- For SAP kernel 3.1I-6.10: kernel patch required, see SAP note 480149
- For SAP kernel 6.20 and above: Included in standard delivery
- It is necessary to activate scripting support in the SAP system, and in the SAP GUI options
- Not possible in ITS environment (SAP GUI for HTML)

11.11.2. SAP Dokumentation

Please see the following SAP notes concerning SAP GUI scripting:

- 480149: Kernel patch requirements
- 587202: Limitations of SAP GUI scripting
- 527737: Composite SAP note on SAP GUI scripting
- 612454: Status and Lifetime
- 619459: SAP GUI Scripting support of SAP applications

The following links provide copies of the original SAP documents, for easy access. Please refer to the original SAP documentation in SAP Service Marketplace (see links below) in order to get the most recent versions of the documents.

SAP GUI Scripting Overview

SAP GUI Scripting User Guide

SAP GUI Scripting Security Settings

SAP GUI Scripting API

SAP Marketplace links to SAP GUI Scripting documentation

Overview

User Guide

Security considerations and requirements

Programming API

Graphical Object Model

11.11.3. GuiXT scripts and SAP GUI scripting

When you execute an ApplyGuiScript command in a GuiXT script, the internal procedure is as follows:

1. SAP GUI receives a new screen description from the application server
2. SAP GUI calls GuiXT. GuiXT reads the GuiXT script and modifies the screen description. It saves the ApplyGuiScript command, but does not yet execute the VB script.
3. SAP GUI builds the full object model from the modified screen description
4. GuiXT calls SAP GUI to execute the specified VBscript

11.11.4. InputScripts and SAP GUI scripting

When you execute an ApplyGuiScript command in an InputScript, please observe the following:

1. A Screen command is necessary before you can work with ApplyGuiScript in the InputScript. It is not (yet) possible to use ApplyGuiScript before the first Screen command. The ApplyGuiScript command is always applied to the preceding Screen in the InputScript.
2. Normally an Enter is user for each Screen in the InputScript. If your VBScript contains a user action that implies an "Enter" (i.e., an application server communication), you have to omit the InputScript Enter command for this screen. Try to avoid such situations, if possible, since the flow of control gets quite complicated if both the InputScript and the VBScript process further screens. In addition, the application server communication invalidates many variables in your VBScript (like screen buttons), so that special care is necessary in writing the VBScript for this case.

The internal procedure is as follows:

1. During InputScript processing, SAP GUI receives a new screen description from the application server

2. SAP GUI calls GuiXT. GuiXT reads the script and modifies the screen description. It saves the ApplyGuiScript commands, but does not yet execute the script.
3. SAP GUI builds the full object model from the modified screen description
4. GuiXT calls SAP GUI to execute the specified VBscript
5. GuiXT performs the Enter action

11.11.5. Writing and testing the VBScript

SAP GUI provides a recording mode that can generate a VBScript directly from your actions. We suggest that you start with such a recorded script. It is a good idea to test the VBScript independently of the GuiXT script, before combining both.

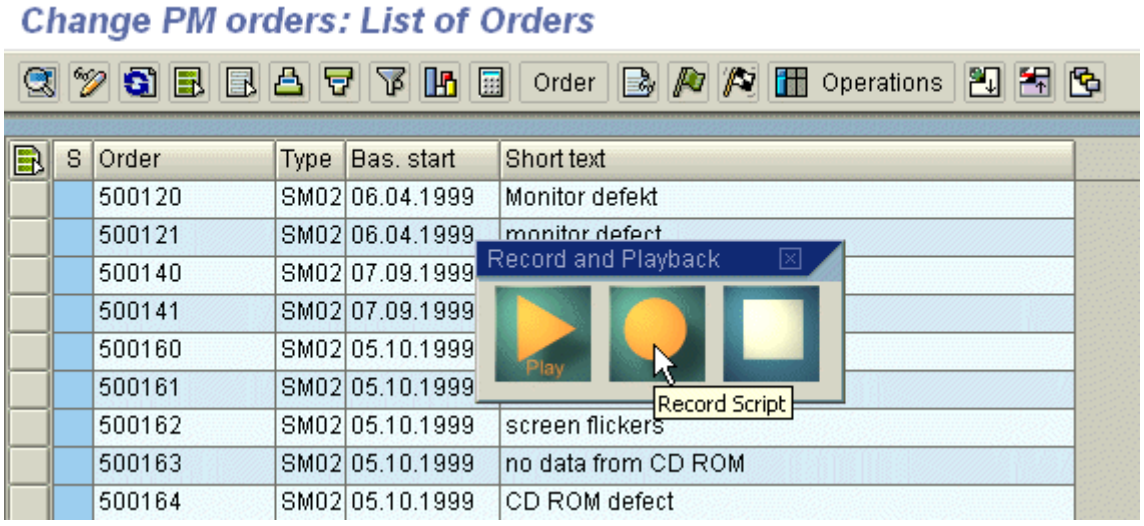
To test the VBScript, display the SAP screen where you want to apply the script. You can either:

- Execute the VBScript outside of SAP GUI by double-clicking on the filenanme, or by entering the name in the MS DOS command prompt
- Or you "drag and drop" the file into the SAP GUI window

We recommend the 2nd possibility (drag&drop), since in this case you use the same SAP GUI interface that GuiXT uses internally. Some VBScript variables, especially the "session" variable, are already defined and preset by SAP GUI when you use drag&drop.

11.11.6. Example: Selecting a fixed row in a grid control

In IW38, you want to select a certain row, say the third row. First, record the action in SAP GUI:



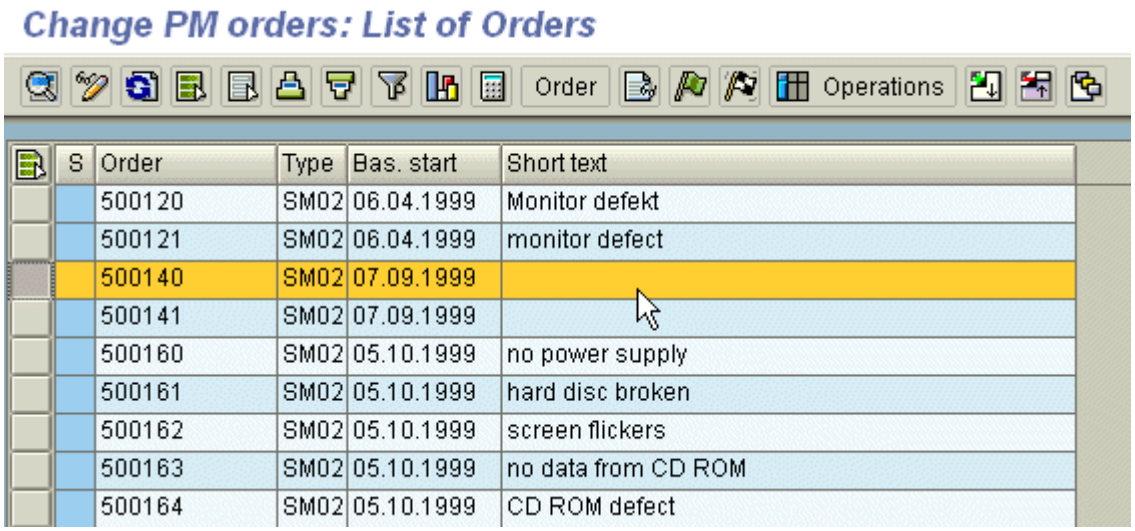
Save the generated VBScript in a suitable directory, e.g. "C:\guixt\sapgui scripts". The script looks as follows:

```
If Not IsObject(application) Then
  Set SapGuiAuto = GetObject("SAPGUI")
  Set application = SapGuiAuto.GetScriptingEngine
End If
If Not IsObject(connection) Then
  Set connection = application.Children(0)
End If
If Not IsObject(session) Then
  Set session = connection.Children(0)
End If
If IsObject(WScript) Then
  WScript.ConnectObject session, "on"
  WScript.ConnectObject application, "on"
End If
session.findById("wnd[0]").resizeWorkingPane 139,15,false
session.findById("wnd[0]/usr/cntlGRID1/shellcont/shell").currentCellRow = 2
session.findById("wnd[0]/usr/cntlGRID1/shellcont/shell").selectedRows = "2"
```

In your GuiXT script, use "ApplyGuiScript" and specify the name of the VBScript file:

```
ApplyGuiScript "C:\guiXT\sapgui scripts\select_grid_row_3.vbs"
```

The screen will then show up like this:



You can simplify the VBScript, if you always use the drag&drop method for testing, since the "session" variable is then defined automatically. The following script does the job as well:

```
Set aw = session.activeWindow()
aw.findById("usr/cntlGRID1/shellcont/shell").selectedRows = "2"
```

Please observe that the row index starts with 0 in SAP GUI scripting, so the third row has index 2.

For more examples please refer to

http://www.synactive.com/docu_d/specials/guiscripting.html

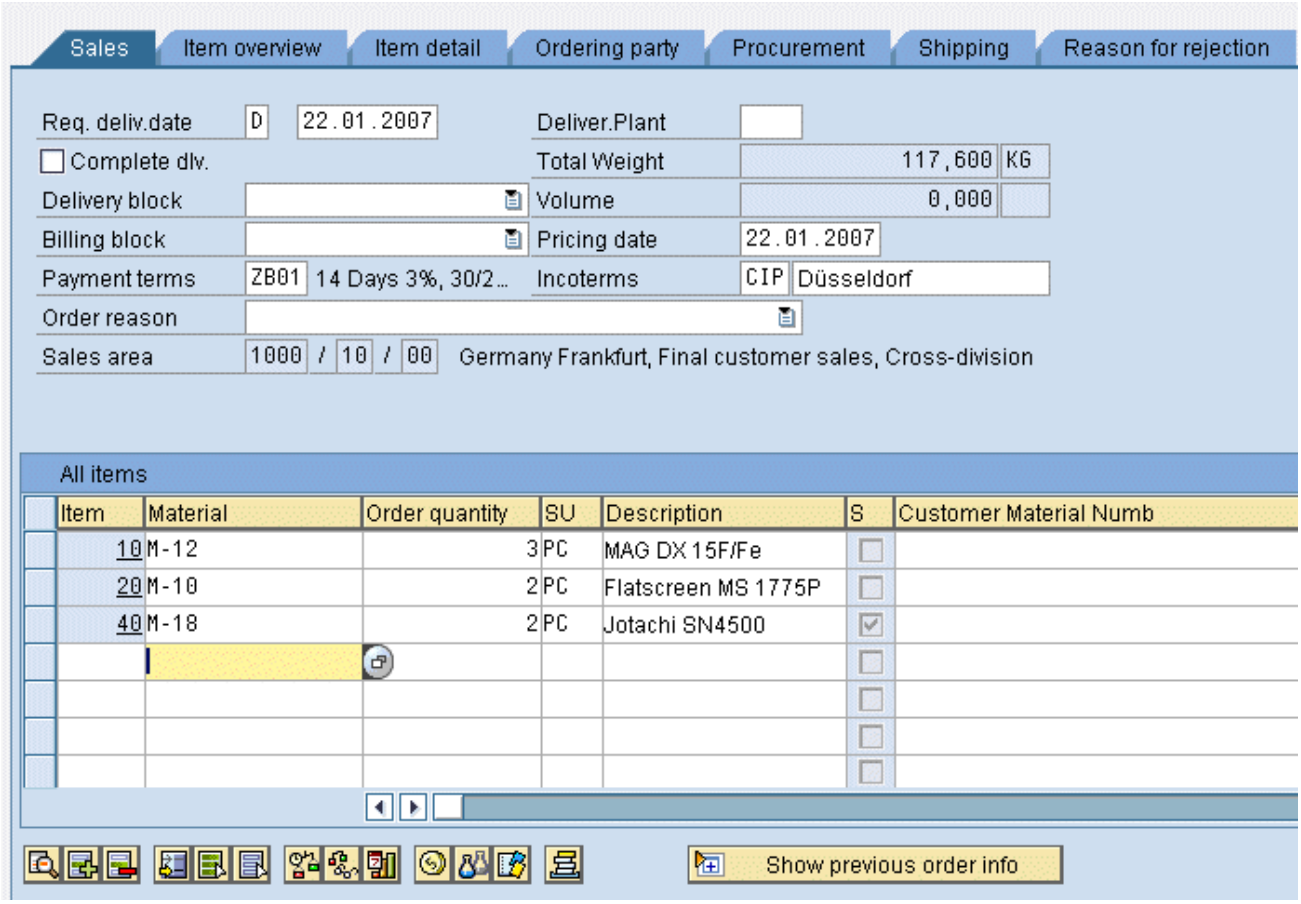
11.12. Aufnehmen von eigenen Spalten in TableControl

Numerous SAP transactions use "table controls" to display data or to let the user enter data in tabular form. With InputAssistant you can create additional columns in a table control, both to display data or to let the user enter additional data to be processed later on in an InputScript.

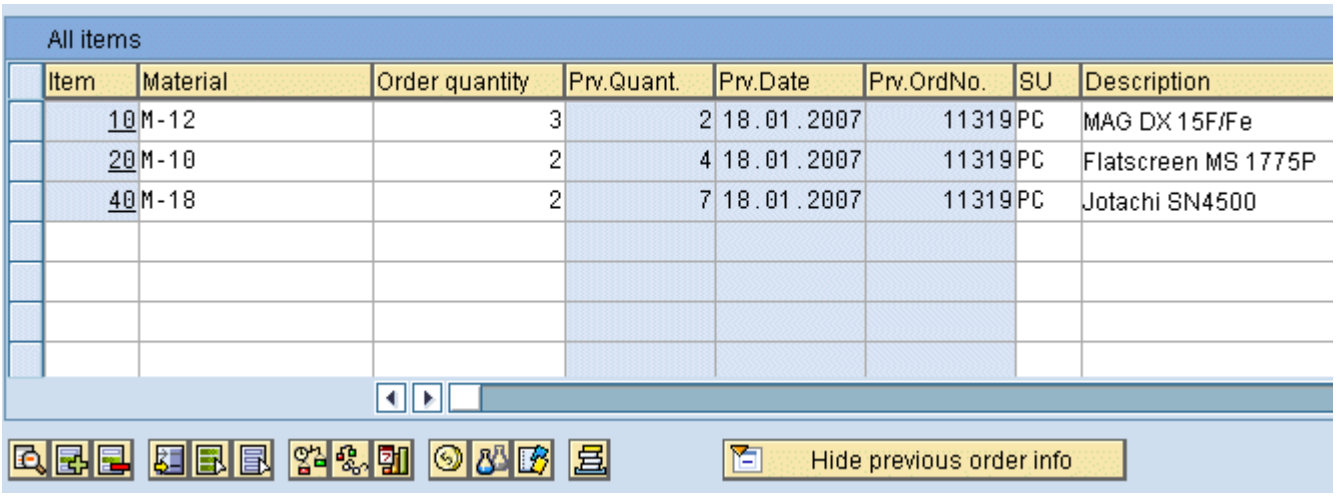
The technique for displaying data is not particularly complex, but handling data entry requires a little more attention to detail. In this tutorial we first cover read-only; then in the second part the scenario of data entry into our new columns.

Our sample application for this tutorial: In transaction VA01 (order entry) we want to offer the user a new button "Show previous order info" that, when clicked, displays the previously ordered quantity for each material entered so far. For each order item we have to read the previous orders for this material from the same customer, using an SAP BAPI call. In the table control we add 3 columns that will be used to display the quantity ordered, the order date and the order number for this material.

For producing the images of the example, we used SAP GUI 7.1 on an SAP ERP 2004 IDES system. The example can also be used, without any changes, on previous SAP versions, e.g. on SAP 4.6C with SAP GUI 6.20 or 6.40.



VA01 Order entry: we add a button "Show previous order info" that will insert three new columns



The previously ordered quantity, order date and order number are shown in three new columns for each material

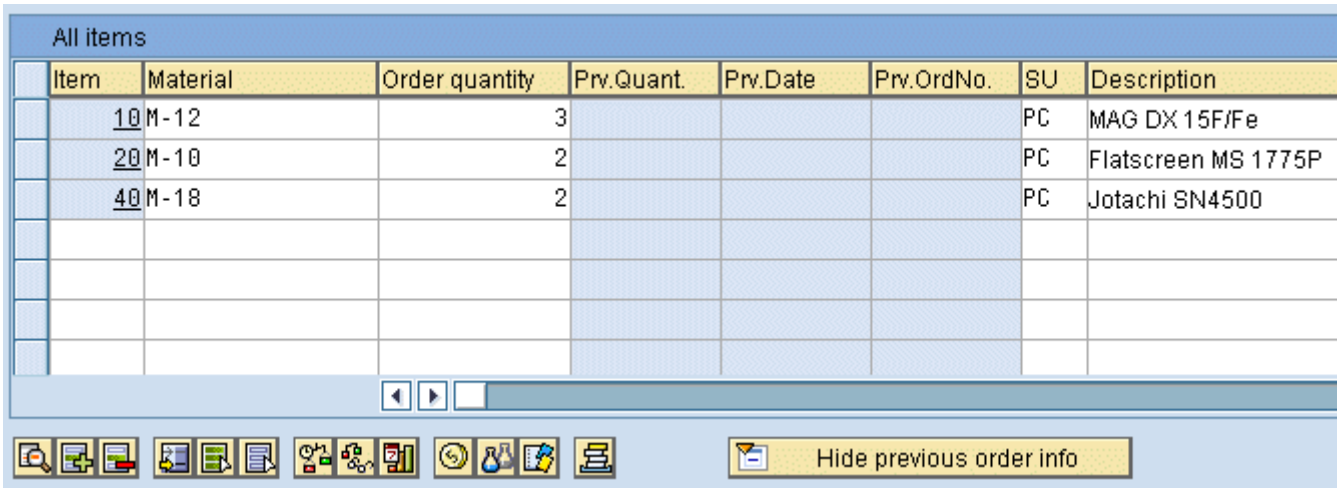
We implement this feature in two steps. First, we implement the new button that displays three new additional table columns:

```
// GuiXT Script SAPMV45A.E4001.TXT

// only for transaction VA01, page sales
if Q[Transaction=VA01] and Q[Page=Sales]

  // show button at end of table (right of iconized button "Item details: ..")
  if not V[VA_show_cols]
    Pushbutton P[Item details: Configuration]+(0,15) "@3S@Show previous order info"
process="VA_show_cols.txt"
  else
    Pushbutton P[Item details: Configuration]+(0,15) "@3T@Hide previous order info"
process="VA_hide_cols.txt"

    // add three new table columns to display previous order quantity for each item
    Column "Prv.Quant." size=10 name="VA_pq"  position=4 -readonly  -alignright
    Column "Prv.Date"   size=10 name="VA_pd"  position=5 -readonly  -alignright
    Column "Prv.OrdNo." size=10 name="VA_po"  position=6 -readonly  -alignright

  endif
endif
```

We can reset the flag V[VA_show_cols] in the inital screen of VA01, otherwise the configuration selected by the user (show or hide the additional info) will remain valid for the next VA01 transaction as well.

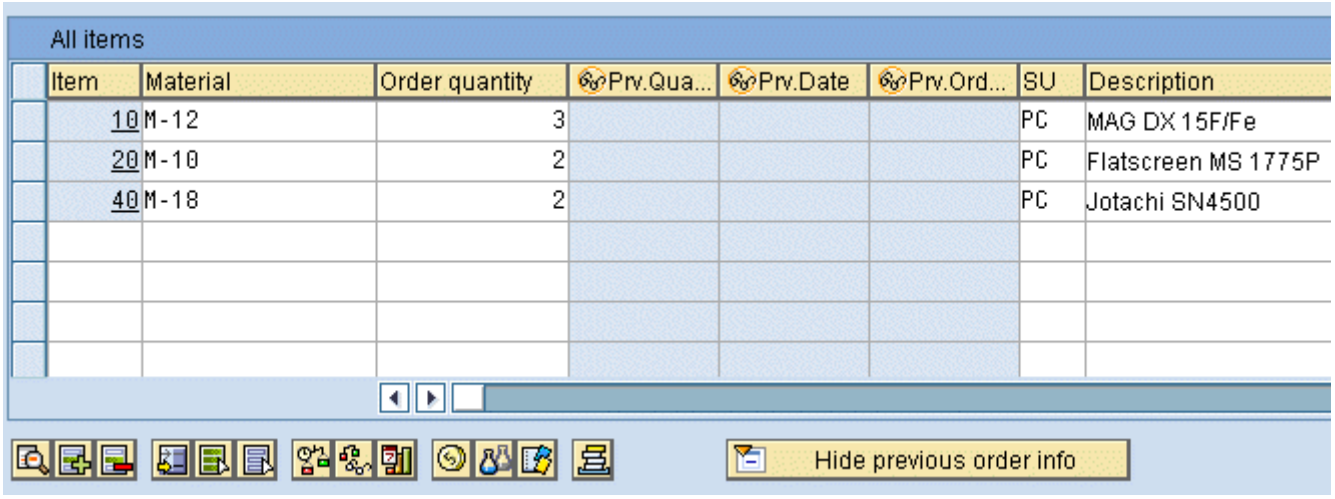
The two InputScripts are quite simple:

```
// InputScript VA_show_cols
Set V[VA_show_cols] "X"
Return

// InputScript VA_hide_cols
Set V[VA_show_cols] ""
Return
```



So far we have created 3 empty columns that the user can show or hide clicking the new button

It is also possible to display a suitable icon in the column headers. Example:



Additional icons are displayed in the column titles

As usual, this is done with the notation "@xx@text..." where xx is the SAP icon id. In this case we used the icon id @OX@.

```
    // add three new table columns with icons to display previous order quantity for
each item
    Column "@OX@Prv.Quant." size=10 name="VA_pq"  position=4 -readonly  -alignright
    Column "@OX@Prv.Date"   size=10 name="VA_pd"  position=5 -readonly  -alignright
    Column "@OX@Prv.OrdNo." size=10 name="VA_po"  position=6 -readonly  -alignright
```

Finally, we need to put the required data into our new columns. The variables behind the columns are named according to the "name=.." parameter in the Column command, and appended with the row number. Example:

row1:	V[VA_pq.1]	V[VA_pd.1]	V[VA_po.1]
row2:	V[VA_pq.2]	V[VA_pd.2]	V[VA_po.2]
row3:	V[VA_pq.3]	V[VA_pd.3]	V[VA_po.3]
row4:	V[VA_pq.4]	V[VA_pd.4]	V[VA_po.4]
row5:	V[VA_pq.5]	V[VA_pd.5]	V[VA_po.5]

© by Synactive GmbH

row6:	V[VA_pq.6]	V[VA_pd.6]	V[VA_po.6]
row7:	V[VA_pq.7]	V[VA_pd.7]	V[VA_po.7]
row8:	V[VA_pq.8]	V[VA_pd.8]	V[VA_po.8]
row9:	V[VA_pq.9]	V[VA_pd.9]	V[VA_po.9]
row10:	V[VA_pq.10]	V[VA_pd.10]	V[VA_po.10]
row11:	V[VA_pq.11]	V[VA_pd.11	V[VA_po.11]

The row index is valid for the whole table control, not only for the visible part. For example, when the user scrolls the table control to row 4, only the rows shown in blue in the above diagram will be visible on screen.

When you set values into the column variables V[VA_pq.1],..., two different approaches are possible. Either you fill all variables at the beginning of the transaction, and do not bother about scrolling. This works fine if the whole table is in readonly mode. In our case it would not be the right approach, since the user can change the material number for each item, or can delete and insert new rows. In such cases it is better to set the column variables each time the screen is displayed, and to set it for the visible (blue coloured) part only. Essentially you need some coding of the following type in order to do this:

```
// loop through all visible table rows
GetTableAttribute T[All items] firstvisiblerow="fvr"

Set V[i] 1         // row number on screen (visible part only)
Set V[k] &V[fvr]   // absolute row number; this would be k=4 in the example above

label prev_order_beg

// valid row left?
Set V[item] "&cell[All items,Item,&V[i]]"
if not V[item]
  goto prev_order_end
endif

//process visible row i. The absolute row number is k

...  Set V[colvar.&V[k]]  "some value"

// next row
Set V[i] &V[i] + 1
Set V[k] &V[k] + 1
goto prev_order_beg

label prev_order_end
```

Finally, we need to read the right data within this framework. For this purpose we call "BAPI_SALESORDER_GETLIST" for each line. The input parameters for the BAPI are the customer number, the material number and the sales area. The BAPI then returns a table of order items for the given customer/material/sales area. The orders are sorted according to date, with the most recent one at the top, so the first line of this table will already contain the most recent order.

Since the order quantity is returned in packed format, we apply the "-unpack" option of the "Set" command. We also delete leading zeros in the order number, and display the date in format DD.MM.YYYY. The whole script is as follows:

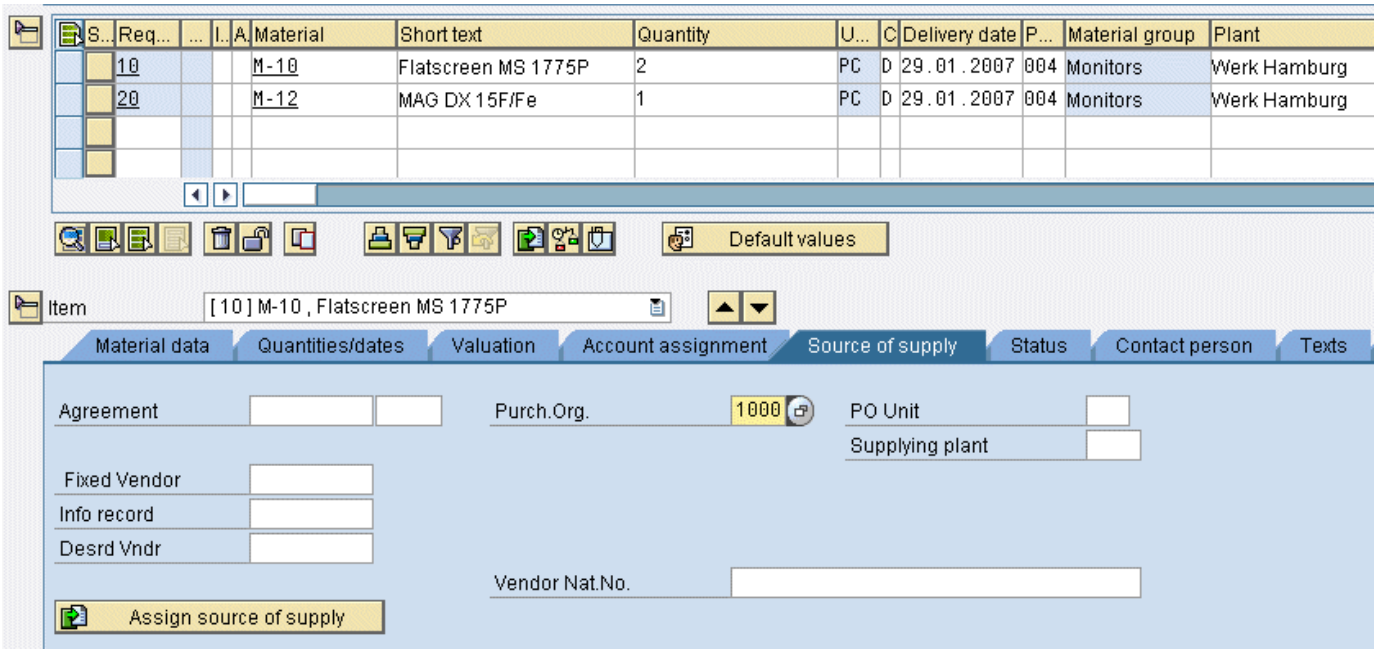
```
// GuiXT Script SAPMV45A.E4001.TXT
```

```
// only for transaction VA01, page sales
if Q[Transaction=VA01] and Q[Page=Sales]

  // show button at end of table (right of iconized button "Item details: ..")
  if not V[VA_show_cols]
    Pushbutton P[Item details: Configuration]+(0,15) "@3S@Show previous order info"
process="VA_show_cols.txt"
  else
    Pushbutton P[Item details: Configuration]+(0,15) "@3T@Hide previous order info"
process="VA_hide_cols.txt"

    // add three new table columns to display previous order quantity for each item
    Column "Prv.Quant." size=10 name="VA_pq"  position=4 -readonly  -alignright
    Column "Prv.Date"   size=10 name="VA_pd"  position=5 -readonly  -alignright
    Column "Prv.OrdNo." size=10 name="VA_po"  position=6 -readonly  -alignright

    // loop through all visible lines
    GetTableAttribute T[All items] firstvisiblerow="fvr"

    Set V[i] 1
    Set V[k] &V[fvr]

    // delete previous values
    Set V[VA_pq.*] ""
    Set V[VA_pd.*] ""
    Set V[VA_po.*] ""

    Set V[customer] "&F[Sold-to party]"
    Set V[customer] 10000000000 + &V[customer]
    Set V[customer] &V[customer](2-11)

label prev_order_beg

    // any item left?
    Set V[item] "&cell[All items,Item,&V[i]]"
    if not V[item]
      goto prev_order_end
    endif

    Set V[matnr] "&cell[All items,Material,&V[i]]"
    if not V[matnr]
      goto prev_order_end
    endif

    // read order info via SAP BAPI
    Call "BAPI_SALESORDER_GETLIST" in.CUSTOMER_NUMBER="&V[customer]"
in.SALES_ORGANIZATION="&F[Sales area]" in.MATERIAL="&V[matnr]" table.SALES_ORDERS=so

    // read first line, it contains the most recent order for this material
    CopyText fromText="so" toString="sowa" line=1

    if Q[ok]   // otherwise there is no previous order for this material

      // now move some fields from the returned BAPI table into our new columns

      // quantity: The BAPI returns it as 'packed decimal' incl. 3 decimal places
      Set V[quantity] "&V[sowa](BAPIORDERS-REQ_QTY)" -unpack
      Set V[quantity] "&V[quantity]" / 1000 decimals=0
      Set V[VA_pq.&V[k]] "&V[quantity]"

      // date: It comes in format YYYYMM, we use DD.MM.YYYY here
      Set V[date] "&V[sowa](BAPIORDERS-DOC_DATE)"
      Set V[VA_pd.&V[k]] "&V[date](7-8).&V[date](5-6).&V[date](1-4)"
```

```
      // order number: comes with leading 0 that we get rid of with a calculation
      Set V[ordno] "&V[sowa](BAPIORDERS-SD_DOC)"
      Set V[ordno] &V[ordno] + 0
      Set V[VA_po.&V[k]] "&V[ordno]"

    endif

    Set V[i] &V[i] + 1
   Set V[k] &V[k] + 1
   goto prev_order_beg

label prev_order_end

  endif  // V[VA_show_cols]

endif // VA01 + page=Sales
```

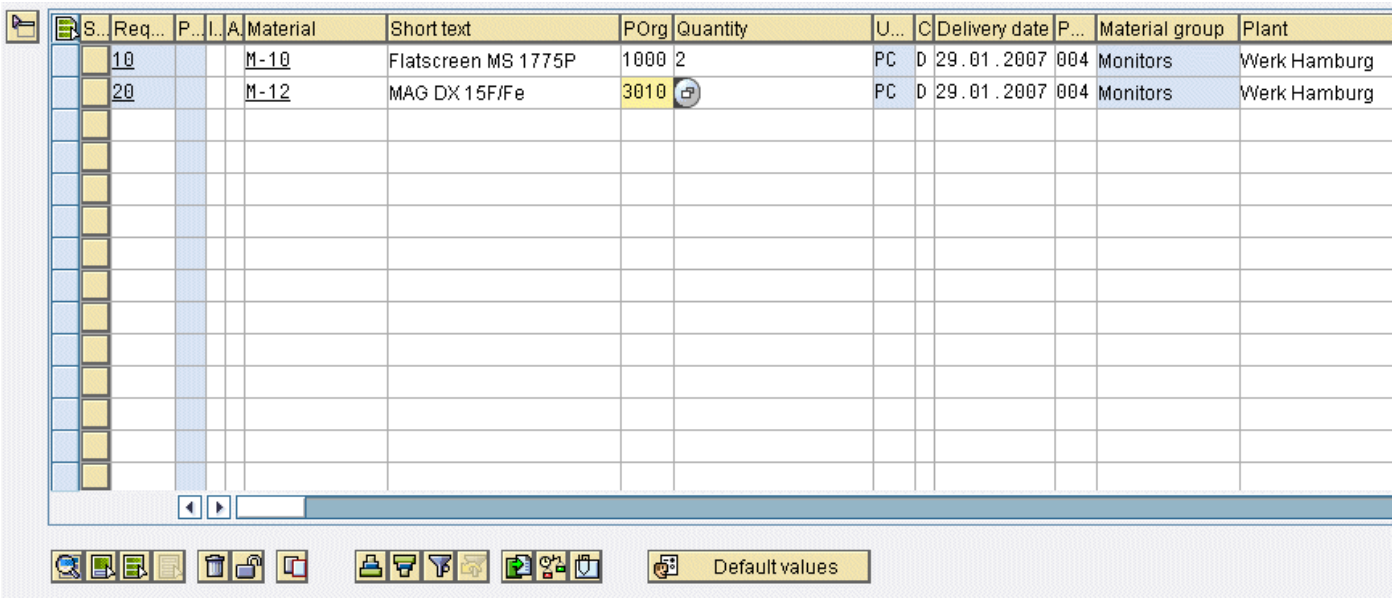
Our second example deals with data entry in transaction ME51N (Create Purchase Requisition) using additional table columns. In this transaction most of the item information can be entered directly in table columns, but some additional fields require that we open the "Detail" view for the item, click on the right tab in the detail view, and enter the information into a separate field on this tab.

Remark: In relatively new SAP systems (e.g. SAP ERP2004 and upwards), the user can choose between a "grid control" and a "table control" for data entry in ME51N (button "Personal setting" in the toolbar). The technique that we describe here applies to the table control only, not to the grid control.



Standard screen: The user enters the "Purch.Org." in the tab "Source of supply" of the detail screen for each item

For one of these fields, the "Purchase Organization" in tab "Source of Supply", we want to create a new column in the table control where the user can enter the purchase organization for each item directly, without having to open up the detail view for this item.


S... Req... P... I.. A Material Short text POrg Quantity U... C Delivery date P... Material group Plant
10 M-10 Flatscreen MS 1775P 1000 2 PC D 29.01.2007 004 Monitors Werk Hamburg
20 M-12 MAG DX 15F/Fe 3010 PC D 29.01.2007 004 Monitors Werk Hamburg
Default values

Enhanced table: The user enters the "Purch.Org." into a new column in the table control.

Compared to other SAP transactions, ME51N has certain idiosyncrasies:

- There is no initial screen, so we have to find a convenient way to reset our variables.
- The user can hide and display various screen parts (text area, item table, detail view).

Another difficulty that we may experience here and in numerous other transactions is that the user can "save" on a popup that is displayed when he presses F3, F12 or F15 (Return, Cancel, Leave) in the transaction. We need a way to start our updating InputScript in these cases as well.

First let's look at the GuiXT script:

```
if Q[Transaction=ME51N]

// if table is visible (can be hidden by user)
if T[Table]

  // no entries in table? then clear all ME51N_ variables
  GetTableAttribute T[Table] FirstVisibleRow=fvr

  if V[fvr=1]
   Set V[matnr] "&cell[Table,Material,1]"
   if not V[matnr]
    Set V[ME51N_*] ""
   endif
 endif

  // create new column
  Column "POrg" position=8 size=4 name="ME51N_EKORG" keycolumn="Requisn. item"
techname="MEREQ3322-EKORG"

  // Additional column is active
  Set V[ME51N_active] "X"
endif
```

```
// Additional column is active? else stop of script processing
if not V[ME51N_active=X]
  Stop
endif

// no entry field in detail screen to avoid synchronization with table column
if F[Purch. organization]
  del F[Purch. organization]
endif

// when saving call own script
on "/11"       process="ME51N_save.txt"


// when checking call own script
on "/39" process="ME51N_save.txt"
  using MODE = "C"


// when the user leaves the transaction, handle the "do you want to save" popup
on "/3"  process="ME51N_leave.txt"
on "/12" process="ME51N_leave.txt"
on "/15" process="ME51N_leave.txt"

endif
```

Some comments on the script:

- We reset the variables when the item table is empty.
- Our new column "POrg" uses the item number as key column. This is necessary since the user can delete items, and the absolute row number will then change for each item above the deleted one. GuiXT uses the value in the key column as index as soon as it is available.
- We also define a searchhelp for the new column (techname=...)
- The standard data entry field for the purchase organization is deleted from the screen. If we were to allow data entry in this field, it would become difficult to synchronize the field with the column data entry
- When the user presses the "check" button, we also first transport the entered "POrg" fields from our added column to the standard SAP fields, so that the checking is done for these fields as well. We use the same script "ME51N_save.txt", but provide an additional parameter MODE = "C".
- For the exit function "/3", "/12", "/15" we start a separate InputScript "ME51N_leave.txt".
- For GuiXT versions before version 2007 Q1 1, it was also necessary to add On "/Menu=x,y" statements in order to handle menu clicks that correspond to /11, /2, /12,... This is now no longer necessary, the On "/11",... is processed automatically in these cases.

The "ME51N_leave.txt" script handles the popup that is displayed when the user leaves the transaction:

```
// InputScript "ME51N_leave.txt"

// popup screen "do you want to save the document first"
```

```
Screen SAPLSPO1.0100
  On "=YES" process="ME51N_save.txt"
    using POPUP = "X"
    using MODE = "S"
```

It also starts the "save" InputScript, setting the parameter POPUP = "X". The "save" InputScript first cancels the popup and then runs through the normal "save" procedure. It also handles cases in which the "detail" view is not yet opened by the user, i.e. the InputScript opens the view and closes it at again the end.

```
// InputScript "ME51N_save.txt"
Parameter POPUP "N" // "X" means: started form leave--popup
Parameter MODE "S" // S= Save C = Check

// No return on error, otherwise entered data could be lost
ProcessingOption ReturnOnError=Off

// A variablle indicates whether the InputScript had to open the "detail view"
Set V[ME51N_detail] ""

// started from "do you want to save?" popup?
if U[POPUP=X]
  Enter "=CANC" // cancel popup, save at end of InputScript
else
  Enter
endif

// Set correct title to be displayed during InputScript processing
if U[Mode=S]
  Set V[ME51N_Title] "Saving the Purchase Requisition... Please wait"
else
  Set V[ME51N_Title] "Checking the Purchase Requisition... Please wait"
endif

Screen saplmegui.0014
  Title "&V[ME51N_Title]"

  // Item table opened?
  if T[Table]
    goto item_table_open
  endif

  // open item table
  Enter "=MEV4001BUTTON"

Screen saplmegui.0014
  Title "&V[ME51N_Title]"

 label item_table_open

  // Row index variables
  Set V[absrow] 1 // Absolute row number
  Set V[relrow] 1 // Relative row number

  // Position details opened?
  if S[REQ_ITEM_DETAIL]
    goto position_details_open
  endif

  // open position details
  Enter "=MEV4002BUTTON"
  Set V[ME51N_detail] "X"

Screen saplmegui.0014
```

```
  Title "&V[ME51N_Title]"

  label position_details_open

  GetTableAttribute T[Table] FirstVisibleRow=FVisRow

  if V[FVisRow=1]
    goto scroll_beg_done
  endif

  // scroll to first line
  Enter "/ScrollToLine=1" Table="T[Table]"

  label new_screen

Screen saplmegui.0014
  Title "&V[ME51N_Title]"

  label scroll_beg_done

  GetTableAttribute T[Table] FirstVisibleRow=FVisRow LastVisibleRow=LVisRow
LastRow=LastRow
  Set V[relrow] 1

  label new_row

  // end of table?
  if V[absrow>&V[LastRow]]
    goto end_of_table
  endif

  // end of screen?
  if V[absrow>&V[LVisRow]]
    Enter "/ScrollToLine=&V[absrow]" Table="T[Table]"
    goto new_screen
   endif

  Set V[item] "&cell[Table,Requisn. item,&V[relrow]]"
  if not V[item]
   goto end_of_table
  endif

  Set V[ME51N_EKORG] "&V[ME51N_EKORG.&V[item]]"
  Set V[ME51N_EKORG_previous] "&V[ME51N_EKORG.&V[item]_previous]"

  // no new input?
  if V[ME51N_EKORG=&V[ME51N_EKORG_previous]]
    Set V[absrow] &V[absrow] + 1
    Set V[relrow] &V[relrow] + 1
    goto new_row
  endif

  // Save input
  Set V[ME51N_EKORG.&V[item]_previous] "&V[ME51N_EKORG]"

  // Set cursor into row and choose detail view
  SetCursor cell[Table,Requisn. item,&V[relrow]]
  Enter "/2"

Screen SAPLMEGUI.0014
  Title "&V[ME51N_Title]"

  if F[Purch. organization]
    goto screen_EKORG
```

```
  endif

  // To TAB "Source of Supply"
  Enter "=TABREQDT7"

Screen SAPLMEGUI.0014
  Title "&V[ME51N_Title]"

  label screen_EKORG

  Set F[Purch. organization] "&V[ME51N_EKORG]"
  Set V[absrow] &V[absrow] + 1
  Set V[relrow] &V[relrow] + 1
  goto new_row

label end_of_table

  // detail view opened?
  if not V[ME51N_detail=X]
   goto screen_collapse_done
  endif

  // collapse detail view again
  Enter "=MEV4002BUTTON"

Screen SAPLMEGUI.0014
  Title "&V[ME51N_Title]"

  label screen_collapse_done

  // Save or check
  if U[MODE=S]
    Enter "/11"
  else
    Enter "/39"
  endif

  // started from popup? else end of script
  if not U[POPUP=X]
   Leave
  endif



// back to main screen after popup + save procedure
Screen SAPLMEGUI.0014
  Enter "/3"

// popup is displayed again (but nothing to save now) -> press "No" to leave the
transaction
Screen SAPLSPO1.0100
  Enter "=NO"
```

When you use "goto" and "label" in the script, please observe the difference between the following versions (the correct one and a wrong one):

(1)

```
Screen saplmegui.0014
// Item table opened?
  if T[Table]
```

```
    goto item_table_open
  endif

  // open item table
  Enter "=MEV4001BUTTON"

Screen saplmegui.0014
   label item_table_open
```

(2)

```
 Screen saplmegui.0014
// Item table opened?
  if T[Table]
    goto item_table_open
  endif

  // open item table
  Enter "=MEV4001BUTTON"

  label item_table_open
Screen saplmegui.0014
```

The second version (label before Screen command) would be wrong: no "Enter" is executed for the first "Screen" command, and therefore the InputScript will display the screen and wait for user input instead of continuing.

Another possible error would be to include a Screen command into an if...endif clause:

```
 Screen saplmegui.0014
// Item table opened?
  if not T[Table]

     // open item table
     Enter "=MEV4001BUTTON"

     Screen saplmegui.0014

  endif
```

Here GuiXT would issue a syntax error message, since if...endif clauses have to be closed within one static Screen block.

After these negative examples, here is an alternative that will work: You use the "goto" to go back to the same Screen command, after entering a suitable function code to open up the table or detail view.

```
...
label main_screen

Screen saplmegui.0014
  Title "&V[ME51N_Title]"

  // Item table opened?
  if not T[Table]
    // open item table
    Enter "=MEV4001BUTTON"
    goto main_screen
  endif
```

```
 // Position details opened?
  if not S[REQ_ITEM_DETAIL]
    // open position details
    Enter "=MEV4002BUTTON"
    goto main_screen
  endif

  GetTableAttribute T[Table] FirstVisibleRow=FVisRow
  if V[FVisRow=1]
    goto scroll_beg_done
  endif

   // scroll to first line
   Enter "/ScrollToLine=1" Table="T[Table]"


label new_screen

Screen saplmegui.0014
  Title "&V[ME51N_Title]"

  label scroll_beg_done

...
```

In this case, it is important that you put the label "main_screen" before the Screen command. If you put it after the Screen command, GuiXT would continue, after the Enter, to execute the script. You would then end up with more than one Enter for one screen, which makes no sense and in fact would lead to a GuiXT syntax error message, and the first "Enter" would be lost. So when you write complex InputScripts please bear in mind that you perform at most one Enter for each Screen, and in most cases exactly one. There are only two cases where it makes sense to have a Screen command without Enter: Either when you want to display the Screen and still stay within the InputScript, a possibility that we used in the "ME51N_leave.txt" InputScript above to handle the popup. Or (a very special case) when you use the "ApplyGUIScript" statement to call up VBScript, and your VBScript program performs an "Enter" action itself (e.g. presses a pushbutton).

© by Synactive GmbH